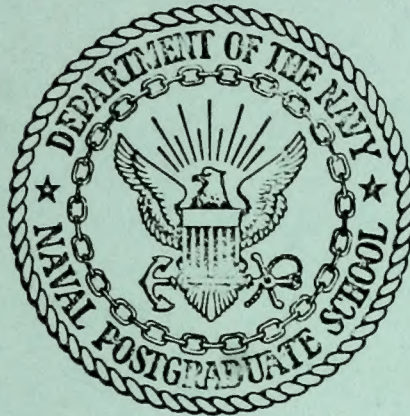


# NAVAL POSTGRADUATE SCHOOL

Monterey, California



## THESIS

NATURAL MODES AND FREQUENCIES OF  
FREE VIBRATION OF SHELLS OF REVOLUTION

—  
ANALYSIS OF A SHELL OF  
LINEARLY VARYING THICKNESS

—  
SATANS-III: THEORY AND USER'S MANUAL

by

Ralph Robert Nebiker

Thesis Advisor:

R.E. Ball

March 1973

Thesis  
N3524

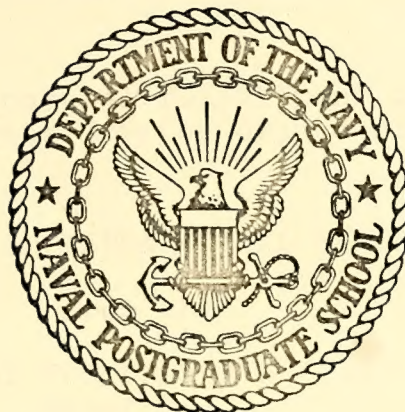
*Approved for public release; distribution unlimited.*

Library  
Naval Postgraduate School  
Monterey, California 93940



# NAVAL POSTGRADUATE SCHOOL

## Monterey, California



# THESIS

NATURAL MODES AND FREQUENCIES OF  
FREE VIBRATION OF SHELLS OF REVOLUTION

—  
ANALYSIS OF A SHELL OF  
LINEARLY VARYING THICKNESS

—  
SATANS-III: THEORY AND USER'S MANUAL

by

Ralph Robert Nebiker

Thesis Advisor:

R.E. Ball

March 1973

*Approved for public release; distribution unlimited.*





Natural Modes and Frequencies of  
Free Vibration of Shells of Revolution

Analysis of a Shell of  
Linearly Varying Thickness

SATANS-III: Theory and User's Manual

by

Ralph Robert Nebiker  
Lieutenant, United States Navy  
B.S., Stevens Institute of Technology, 1966

Submitted in partial fulfillment of the  
requirements for the degree of

MASTER OF SCIENCE IN AERONAUTICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL  
March 1973





ABSTRACT

A digital computer program known as SATANS-III is developed as a modification to the program SATANS-I, incorporating the capability to perform free vibration analyses for the natural frequencies and modes of thin, elastic shells of revolution. The geometric and material properties of the shell may vary along the meridian. SATANS-III retains the SATANS-I capability of performing geometrically nonlinear static and dynamic analyses of arbitrarily loaded shells. The free vibration analysis employs the inverse iteration method with spectral shifts, including the capability to search a specific range of frequencies. Several example problems are solved to illustrate the program's validity and accuracy. A user's manual for SATANS-III is presented. In addition, the problem of handling a large, strongly banded, unsymmetric and occasionally singular stiffness matrix in conjunction with a singular mass matrix is treated. A discussion of several solution procedures with respect to this problem is given.





## TABLE OF CONTENTS

I.	INTRODUCTION -----	12
II.	BRIEF DESCRIPTIONS OF SATANS: VERSIONS I, II AND III -----	14
	A. THE ORIGINAL SATANS -----	14
	B. SATANS-I -----	15
	C. SATANS-II -----	16
	D. SATANS-III -----	16
III.	SOLUTION PROCEDURE -----	18
	A. SHELL COORDINATE SYSTEM -----	18
	B. METHOD OF SOLUTION -----	18
	C. STATIC ANALYSIS -----	28
	D. DYNAMIC ANALYSIS -----	30
IV.	DEVELOPMENT OF SATANS-III -----	31
	A. FREE VIBRATION EQUATIONS -----	31
	B. CHARACTERISTICS OF THE PROBLEM -----	32
	1. Desired Results -----	32
	2. Nature of the Problem -----	32
	C. SOLUTION TO THE EIGENVALUE PROBLEM -----	34
	D. TESTING OF SATANS-III -----	40
V.	CONCLUSIONS -----	42
	APPENDIX A OTHER SOLUTION METHODS -----	44
	APPENDIX B NATURAL FREQUENCIES OF SOME SELECTED SHELLS -----	52
	APPENDIX C CONVERTING SATANS-I TO SATANS-III PART ONE: COMMON BLOCK MODIFICATIONS ----	63
	PART TWO: MODIFYING SATANS-I SUBROUTINES -----	71





PART THREE:	NEW SUBROUTINES - THEORY AND FLOWCHARTS -----	76
APPENDIX D	EXPERIENCE AND RECOMMENDATIONS ON USING SATANS-III -----	169
APPENDIX E	SATANS-III INPUT CARD DATA GUIDE -----	172
APPENDIX F	INPUT AND OUTPUT PART ONE: USER SUPPLIED SUBROUTINES -----	182
	PART TWO: SAMPLE PRINTOUT -----	184
APPENDIX G	EXAMPLES OF USER SUPPLIED SUBROUTINES AND A LISTING OF SATANS-III -----	190
LIST OF REFERENCES	-----	298
INITIAL DISTRIBUTION LIST	-----	301
FORM DD 1473	-----	302





## LIST OF TABLES

I.	Minimum Natural Frequencies of a Thin Circular Cylindrical Shell -----	52
II.	Natural Frequencies of a Thin Circular Cylindrical Shell -----	53
III.	Minimum Natural Frequencies of a Freely Supported Circular Cylindrical Shell with and without Inplane Inertia -----	54
IV.	Minimum Natural Frequencies of a Shell of Positive Gaussian Curvature -----	56
V.	Effect of a Linearly Varying Thickness on Minimum Frequencies -----	58
VI.	Shell Configurations -----	60
VII.	Boundary Condition Specification -----	62
VIII.	Contents of SATANS-III Common Blocks -----	64
IX.	Some FORTRAN Variables -----	69



## LIST OF FIGURES

1.	Shell Geometry and Coordinate System -----	19
2.	Positive Directions for Variables -----	20
3.	System Stiffness Matrix, Load and Solution Vectors -----	26
4.	Nonlinear Behavior Affecting Convergence -----	29
5.	Reduction of a Matrix to Hessenberg Form -----	46
6.	Minimum Natural Frequencies of a Freely Supported Circular Cylindrical Shell -----	55
7.	Minimum Natural Frequencies of Three Cones -----	57
8.	Effect of a Linearly Varying Thickness on Minimum Natural Frequencies -----	59
9.	Search Scheme Schematic -----	93
10.	Subroutine VIBERS(*) Flowchart -----	97
11.	Subroutine ITREAL(*) Flowchart -----	102
12.	Subroutine ITRATE Flowchart -----	105
13.	Subroutine ITCPLX(*) Flowchart -----	121
14.	Subroutine RANGER(*) Flowchart -----	127
15.	Subroutine ADJUST(IADJ,*,*) Flowchart -----	134
16.	Subroutine BTM(*) Flowchart -----	144
17.	Subroutine TOPPER(*) Flowchart -----	151
18.	Subroutine INITAA Flowchart -----	158
19.	Subroutine START(K) Flowchart -----	163
20.	Subroutine VECOUT(IND,*) Print-out Samples -----	166





# TABLE OF SYMBOLS

$a$	characteristic shell reference length
$b$	nondimensional inplane stiffness
$d$	nondimensional bending stiffness
$E_y$	Young's modulus
$E_o$	Reference Young's modulus
$f$	natural frequency, CPS or Hz
$h$	shell thickness
$h_o$	shell reference thickness
$h_1$	thickness at initial edge
$h_2$	thickness of final edge
$ht$	shell height at axis of revolution
$M_s, M_\theta, M_{s\theta}$	bending and twisting moments per unit length
$m$	shell mass density, lb-sec <sup>2</sup> /in <sup>4</sup>
$m_s, m_\theta, m_{s\theta}$	nondimensional bending and twisting moment series coefficients
$N, n$	Fourier series circumferential harmonic (wave number)
$N_s, N_\theta, N_{s\theta}$	membrane forces per unit length
$\hat{N}_{s\theta}$	effective shear force
$p$	nondimensional pressure load series coefficient
$\hat{Q}_s$	effective transverse force
$q, q_s, q_\theta$	normal, meridional and circumferential components of applied pressure loading
$R_s, R_\theta$	principal radii of curvature
$r$	normal distance from the axis of shell



$\bar{r}$	normal distance from shell axis used in nondimensional frequency parameter $\bar{E}$
$r_1$	normal distance from shell axis at initial edge
$r_2$	normal distance from shell axis at final edge
$s$	meridional shell coordinate
$S_T$	total meridional shell length
$T$	dimensional time
$T_0$	reference time
$t$	nondimensional time
$t_s, t_\theta, t_{s\theta}$	nondimensional membrane force series coefficients
$t_T$	nondimensional thermal membrane force series coefficients
$U, V, W$	displacements tangent to the meridian, tangent to the parallel circle and normal to the reference surface
$u, v, w$	nondimensional series coefficients of $U, V,$ and $W$
$\alpha^r$	$r^{\text{th}}$ iteration Rayleigh quotient eigenvalue, $\frac{-1}{(\omega^r T_0)^2}$
$\beta$	spectral shift point
$\gamma$	$\rho'/\rho$
$\Delta$	distance between finite difference stations = $\frac{S_T}{(a(KMAX - 1))}$
$\delta t$	nondimensional time interval
$\zeta$	shell coordinate perpendicular to reference surface
$\theta$	shell circumferential coordinate





$\mu_1$	nondimensional mass at $i^{\text{th}}$ station  $= \frac{a^2}{h_o^2 E_o^2 T_o^2} \int m \, d\zeta$
$\Xi$	nondimensional frequency parameter  $= \omega r \sqrt{\frac{m(1-\nu^2)}{E_y}}$
$\nu$	Poisson's ratio
$\xi$	nondimensional meridional coordinate, $s/a$
$\rho$	nondimensional normal distance to shell axis, $r/a$
$\sum_{i=m,j}^n$	summation over $i$ from $m$ to $n$ in increments of $j$
$\sigma_o$	reference stress level
$\Phi, \Phi_s, \Phi_\theta$	reference surface rotations
$\phi, \phi_s, \phi_\theta$	nondimensional series coefficients for reference surface rotations
$\omega$	natural frequency, radians per second
$\omega_s, \omega_\theta$	nondimensional curvatures, $a/R_s, a/R_\theta$

#### MATRIX NOTATION

$A, B, \bar{B}, C, E, F, G, \bar{G}, H, J, P, DEE, DST$	$4 \times 4$ matrices. Elements of the matrices are defined in Refs. 1 and 2.
$\bar{e}, \bar{f}, g, \bar{g}$	$1 \times 4$ load vectors. Elements are defined in Refs. 1 and 2.
$K$	system stiffness matrix
$\ell$	$1 \times 4$ boundary condition vector
$M$	system mass matrix
$W^r$	system solution vector before eigenvalue extraction



$z$	1 x 4 solution vector
$z^r$	system solution vector after scaling and eigenvalue extraction.
$\bar{\Lambda}, \bar{\Omega}$	4 x 4 boundary condition matrices
$\bar{\mu}$	4 x 4 mass matrix
$[ \ ]$	m x n matrix
$\{ \}$	column vector
$[ \_ ]$	row vector
$  \  $	absolute value of a scalar
$i=m(k)n$	i is stepped from m to n in increments of k

#### SUPERSCRIPTS

$r$	$r^{\text{th}}$ iteration
$T$	transpose of a matrix or vector
$'$	(prime) partial differentiation in the meridional direction
$-1$	inverse of a matrix





### ACKNOWLEDGMENT

The author wishes to express his deepest gratitude to Dr. Robert E. Ball for his abiding faith, trust and understanding. Dr. Ball's insight and expert professional guidance were principle factors leading to the successful completion of this project.



## I. INTRODUCTION

The free vibration analysis of shells of revolution is of major interest to the modern structural analyst since shells are used in the construction of everything from spacecraft to nuclear reactors. The modes of free vibration and the associated natural frequencies are of fundamental importance in understanding the dynamic behavior of shells.

This thesis is devoted to the incorporation of a free vibration analysis capability in the computer program known as SATANS (Static And Transient Aalysis, Nonlinear, Shells). The development of SATANS is described in Refs. 1, 2 and 3. In Section II of this thesis a brief description and history of the three versions of SATANS (known as SATANS, SATANS-I and SATANS-II) is presented. In Section III a brief explanation of the solution procedure used in SATANS is given. In Section IV, SATANS-III is developed. SATANS-III is a modification of SATANS-I incorporating the capability of a free vibration analysis. Major consideration was given to: (1) matching the vibration analysis procedure to the present program without the necessity of reformulating the original program, (2) maintaining the capability of having the shell material and geometric properties vary along the meridian, (3) having assurance of completely covering a frequency range in a given





Fourier harmonic. The selection of the inverse iteration procedure with spectral shifts meets these goals. In addition, a backup capability to handle unusual situations, such as a pair of complex conjugate eigenvalues, is included.

To illustrate the validity of SATANS-III the results of six analyses of shells of uniform thickness are compared with published results in Appendix B. In addition new results showing the analysis of a shell of linearly varying thickness is also given in Appendix B.

In Appendix A there is a short discussion of the difficulties of the present problem of a large, strongly banded, unsymmetric and occasionally singular stiffness matrix in conjunction with a singular mass matrix using other solution procedures.

The coding details for converting SATANS-I to SATANS-III are given in Appendix C. Appendices D through G constitute the user's manual for SATANS-III.



## II. BRIEF DESCRIPTIONS OF SATANS: VERSIONS I, II AND III

SATANS [Refs. 1,2] existed in three forms prior to this work; the original code and two later modifications, SATANS-I and SATANS-II [Ref. 3].

### A. THE ORIGINAL SATANS

SATANS [Refs. 1,2] is a digital computer program capable of performing geometrically nonlinear static and dynamic analyses of arbitrarily loaded shells of revolution. It is based upon a combination of Fourier series for the circumferential behavior and finite differences for the meridional analysis. The conditions that the program is based upon are:

1. The shell material is isotropic and Poisson's ratio is constant.
2. The elastic modulus may vary through the shell thickness.
3. All material and geometric properties are axisymmetric and may vary along the shell meridian.
4. The boundaries may be free, elastically restrained, fixed or closed (pole).
5. The loading may be any combination of a temperature distribution, pressure and edge load.
6. The applied pressure and temperature distributions, and initial conditions in the case of a dynamic analysis, must be symmetric about a reference meridional plane.





7. The maximum number of circumferential Fourier harmonics, MAXM, used to describe the loading and solution, is 10. The maximum number of meridional finite difference stations, KMAX, is 200. The limit on the product of KMAX and MAXM is 200.

SATANS requires approximately 155 K-bytes of core storage on an IBM 360/67 using the FORTRAN-H compiler with optimization level 2.

#### B. SATANS-I

SATANS-I, developed by Ryan [Ref. 3], consists mainly of a rearrangement of the original program and the addition of an equipment independent plotting package capable of handling 100 points. The rearrangement allows approximately 99% of the code to be pre-compiled and linked internally and stored on rapid-access secondary storage (disk, cell, tape, drum, etc). This stored package is called by a simple six card main program. The plotting package enables the user to obtain plots of all input and output variables on the high speed line printer. In addition, solutions at 36, vice 6 for SATANS, meridians may be printed and/or plotted. The restrictions on KMAX and MAXM are the same as for SATANS.

SATANS-I requires approximately 206 K-bytes of storage on an IBM 360/67 using the FORTRAN-H compiler with optimization level 2. Execution time is increased over that of SATANS by .01 seconds for each 100 point plot generated.



### C. SATANS-II

SATANS-II, also developed by Ryan [Ref. 3], is an extensive expansion of SATANS-I. The following improvements were incorporated:

1. The ability to handle arbitrary initial imperfections in the shape of the shell was introduced.
2. The restriction on the applied loads, that they be symmetric about a reference meridian, was removed.
3. The maximum number of circumferential Fourier harmonics describing the completely general loads and the solution was increased to 13, and the maximum number of meridional stations was reduced to 100. The restriction on the product of KMAX and MAXM was raised to 951.

SATANS-II is a large program requiring 426 K-bytes of main core on an IBM 360/67 using the FORTRAN-H compiler with optimization level 2.

### D. SATANS-III

SATANS-III is a modification of SATANS-I which incorporates the ability to perform a free vibration analysis of shells of revolution. SATANS-III operates under the same conditions as SATANS and SATANS-I. The procedure used to determine the modes and frequencies is inverse iteration with spectral shifts. One of three modes of operation is specified by the user. The three modes are:

1. Determination of the lowest frequency in a user specified circumferential harmonic.
2. Determination of the frequency closest to a user specified frequency in a user specified circumferential harmonic (supplied shift point mode).





3. Frequency range search (circumferential harmonic and end points of the frequency range specified by the user).

The assumptions made in determining the natural modes and frequencies are:

1. No loads of any type.
2. Small deflections and rotations.
3. All nonlinear effects are ignored.

SATANS-III requires 234 K-bytes of main core on an IBM 360/67 using the FORTRAN-H compiler with optimization level 2.



### III. SOLUTION PROCEDURE

The solution procedure used in all of the versions of SATANS is based on Sanders' field equations [Ref. 4], with translational inertia terms added, assuming small strains and moderately small, or small, rotations. The basic method of solution was developed by Budiansky and Radkowski [Ref. 5] for the linear static case. A brief presentation of the method of solution is presented here for continuity. For a more detailed explanation refer to Refs. 1 and 2.

#### A. SHELL COORDINATE SYSTEM

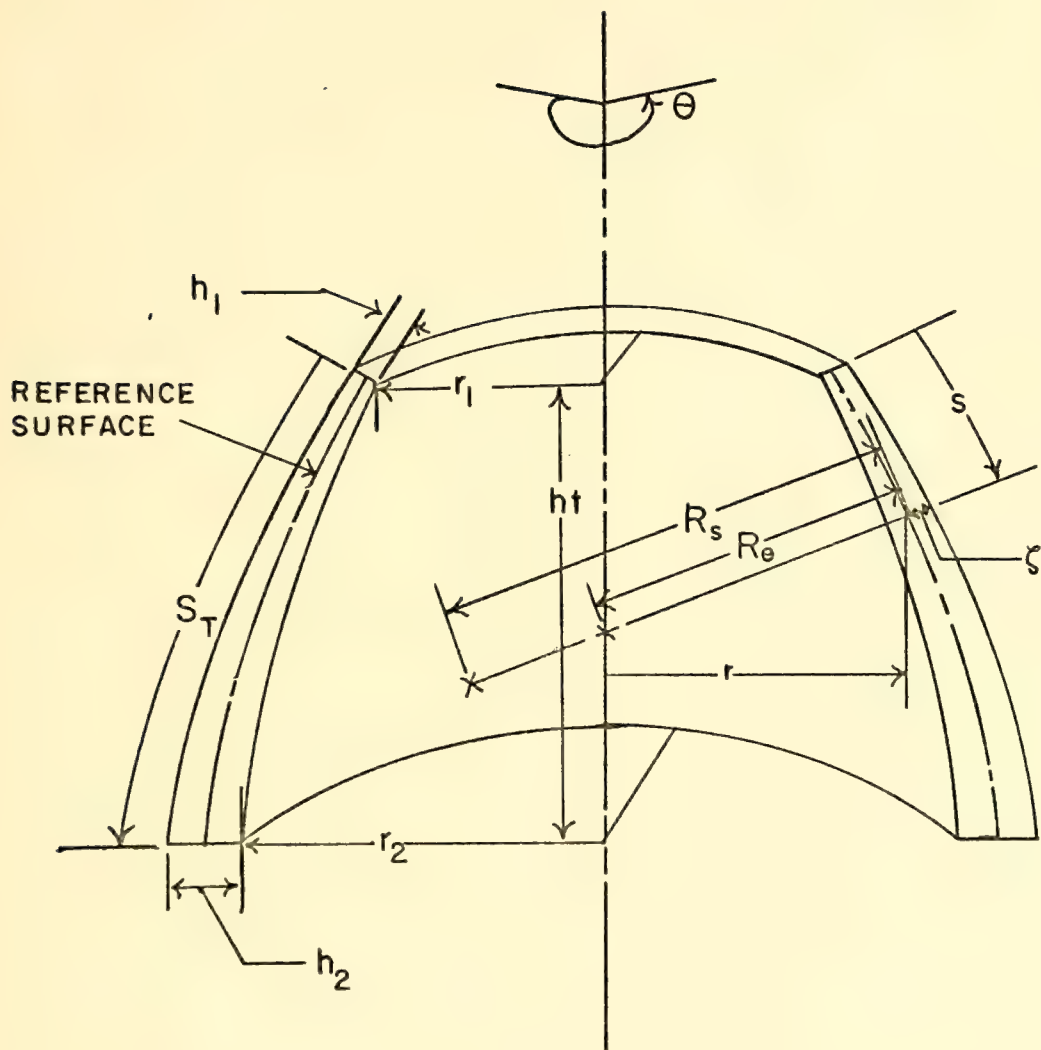
Figure 1 depicts the orthogonal coordinate system used to describe the shell geometry.  $s$  is the distance along the meridian on the reference surface,  $\theta$  is the circumferential angle, measured from some reference meridian, about the shell axis of revolution, and  $\zeta$  is the coordinate through the thickness of the shell measured normal to the reference surface. The integral  $\int E_y \zeta d\zeta = 0$  positions the reference surface, where  $E_y$  is Young's modulus.

Figure 2 depicts the positive directions for the displacements, rotations, forces, moments and loads.

#### B. METHOD OF SOLUTION

In the SATANS-I analysis the dependent variables in Sanders' nonlinear field equations are expanded into either a Fourier sine or cosine series in terms of  $n\theta$ , where  $n$  is





SHELL GEOMETRY AND COORDINATE SYSTEM

FIGURE 1





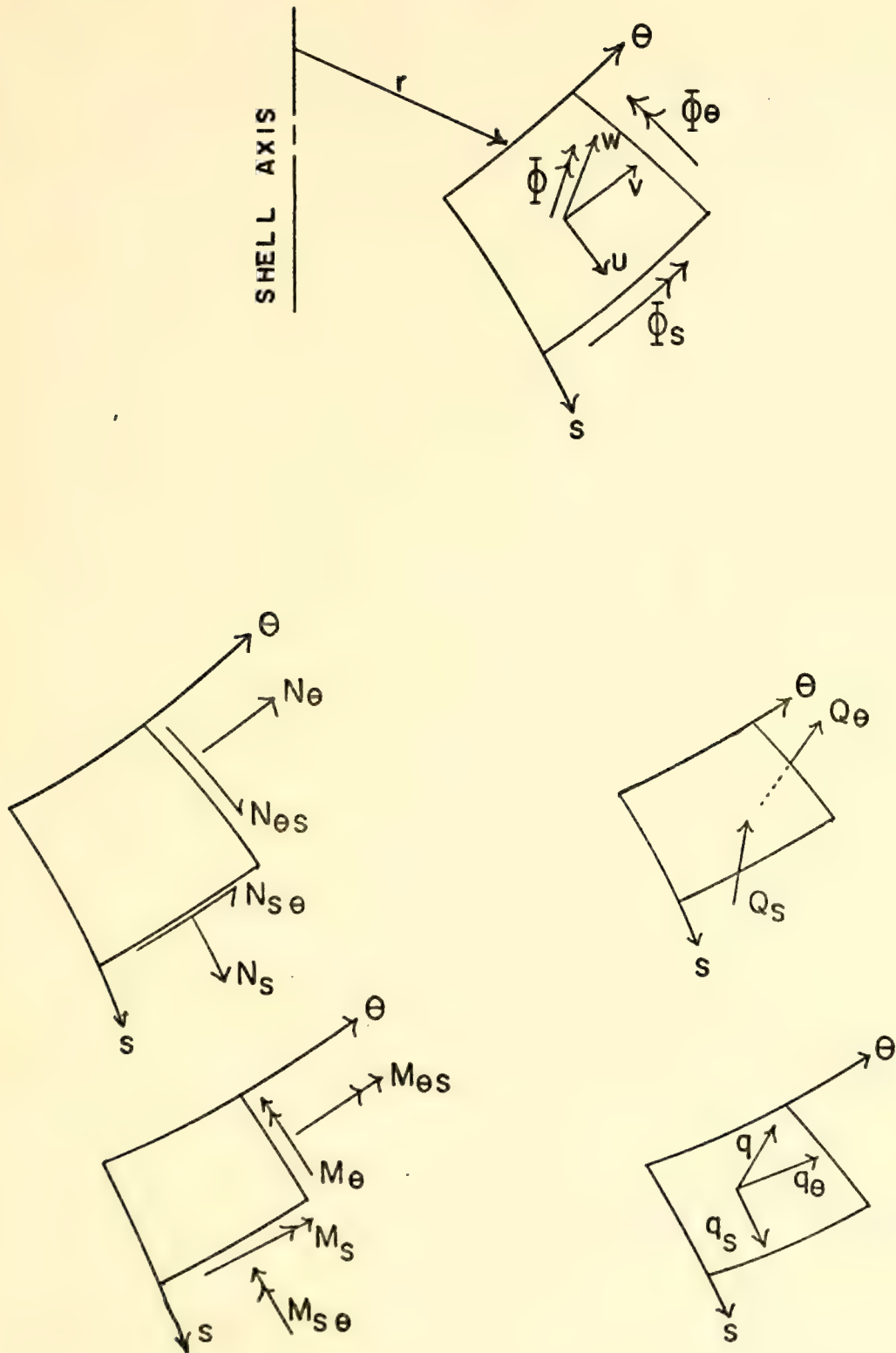


FIGURE 2 POSITIVE DIRECTIONS  
FOR VARIABLES



the Fourier harmonic index. This results in a complete set of governing equations for each Fourier harmonic  $n$ . The nonlinear terms are products of two series. These products are expressed as a single series, the coefficients of which are also series. In order to uncouple the sets of equations the "pseudo load" approach is used, i.e. the nonlinear coupling terms are treated as known quantities and are included with the loads on the right side of the equations. Thus the left-hand side of the equations is linear and the sets are uncoupled.

Following the method suggested in Ref. 5 each set of uncoupled equations (one set for each value of  $n$ ) is expressed as four second order differential equations. The independent variables are the series coefficients  $u^{(n)}$ ,  $v^{(n)}$ ,  $w^{(n)}$  and  $m_s^{(n)}$ , representing the nondimensional meridional, circumferential and normal displacements and the meridional bending moment respectively. For convenience the superscript  $n$  will be dropped henceforth.

For each value of  $n$ , the set of four second order differential equations may be written in matrix notation as

$$E z'' + F z' + G z = \bar{e} + \bar{\mu} \frac{\partial^2 z}{\partial t^2} \quad (1)$$

where  $t$  is nondimensional time. Primes denote partial differentiation with respect to the nondimensional



meridional coordinate  $\xi$ . Vector  $\bar{e}$  contains the load and nonlinear terms.

The solution vector  $z$  is

$$z = \begin{Bmatrix} u \\ v \\ w \\ m_s \end{Bmatrix}$$

The mass matrix  $\bar{\mu}$  is

$$\bar{\mu} = \mu \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

where the nondimensional mass  $\mu$  is defined by the integral

$$\mu = \frac{a^2}{h_o E_o T_o^2} \int m d\xi \quad (2)$$

where  $m$  is the mass density of the shell material and  $a$ ,  $h_o$ ,  $E_o$ , and  $T_o$  are nondimensionalizing parameters. The elements of the  $4 \times 4$  matrices  $E$ ,  $F$  and  $G$  and the vector  $\bar{e}$  are given in Ref. 2.

Equations (1) are solved numerically using spatial and timewise differencing schemes. In both the static and dynamic analysis the first and second derivatives of  $z$





are approximated using the central finite difference schemes

$$z_i' = (z_{i+1} - z_{i-1})/\Delta$$

$$z_i'' = (z_{i+1} - 2z_i + z_{i-1})/\Delta^2$$

where  $i$  denotes the station along the meridian and  $\Delta$  is the uniform spacing between stations.

Houbolt's backward differencing scheme is used to approximate the second derivative with respect to time for the dynamic case.

$$\left. \frac{\partial^2 z}{\partial t^2} \right)_{i,j} = \frac{(2z_{i,j} - 5z_{i,j-1} + 4z_{i,j-2} - z_{i,j-3})}{(\delta t)^2}$$

The time step is denoted by  $j$  and  $\delta t$  is the nondimensional time interval.

The resulting set of four second order difference equations can be written in the matrix form

$$A_i z_{i+1,j} + \bar{B}_i z_{i,j} + C_i z_{i-1,j} = \bar{g}_{i,j} \quad (3)$$

for  $i = 1, 2, 3, \dots, KMAX$ . The  $4 \times 4$  matrices  $A$ ,  $\bar{B}$  and  $C$  and the vector  $\bar{g}$  are given by

$$A_i = (2E_i/\Delta) + F_i$$

$$\bar{B}_i = (-4E_i/\Delta) + 2\Delta\bar{G}_i$$



$$C_1 = (2E_1/\Delta) - F_1$$

$$\bar{G}_1 = G_1 - (2\bar{\mu}_1/(\delta t)^2)$$

$$\bar{g}_{1,j} = 2\Delta\bar{e}_{1,j} + \frac{2\Delta\bar{\mu}_1}{(\delta t)^2} (-5z_{1,j-1} + 4z_{1,j-2} - z_{1,j-3})$$

In the static analysis  $j$  denotes the load step and  $\mu_1 = 0$ .

The boundary conditions on  $z$  are entered on cards as the elements of two  $4 \times 4$  matrices,  $\bar{\Omega}$  and  $\bar{\Lambda}$ , and the column vector  $\ell$ . The boundary conditions specify either the forces or the displacements at the shell boundary for all  $n$  considered.

The boundary condition matrix equation is

$$\bar{\Omega} \begin{Bmatrix} N_s \\ \hat{N}_s \\ \hat{Q}_s \\ \phi_s \end{Bmatrix} + \bar{\Lambda} \begin{Bmatrix} U \\ V \\ W \\ M_s \end{Bmatrix} = \ell$$

where all terms are identified in Figure 2. The  $\hat{\phantom{x}}$  above  $\hat{N}_s$  and  $\hat{Q}_s$  denotes the effective edge variables. As an example, a clamped edge is specified by

$$\bar{\Omega} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \bar{\Lambda} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$



and

$$\ell = \begin{Bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{Bmatrix}$$

In order to convert the boundary conditions to finite difference form, the derivatives with respect to the meridional coordinate in the forces and rotations are replaced by the central finite difference schemes.

For each value of  $n$  the system of  $KMAX + 2$  matrix equations may be written as

$$K Z = Y \quad (4)$$

Figure 3 depicts the elements of the  $K$  matrix and the  $Z$  and  $Y$  vectors.\* The first and last rows of  $K$  are the boundary condition equations. The elements of the  $4 \times 4$  matrices  $H$  and  $J$  and the four element vector  $\bar{F}$  are given

---

\*There are  $KMAX$  stations over the length of the shell where Equation (3) applies. The first and last stations are at the initial and final edges. This leads to  $KMAX$  equilibrium equations. The boundary equations are also applied at the initial and final edges of the shell. Thus there are  $KMAX + 2$  matrix equations. The application of Equation (1) and the boundary equations at the edges of the shell introduce one imaginary station off each edge of the shell. This results in  $KMAX + 2$  unknown values of  $z$ . The vector  $Y$  consists of the  $KMAX$  values of  $\bar{g}$  and the two terms at the initial and final positions representing the loads and the boundary condition values at the edges.





$$\begin{array}{c}
 \left[ \begin{array}{ccc}
 \frac{-\Omega_1 H_1^N}{2\Delta} & \Omega_1 J_1^N & \frac{\Omega_1 H_1^N}{2\Delta} \\
 C_1^N & \bar{B}_1^N & A_1^N \\
 & C_2^N & \bar{B}_2^N & A_2^N \\
 & & \circ & \circ & \circ \\
 \hline
 & \circ & \circ & \circ \\
 & C_K^N & \bar{B}_K^N & A_K^N \\
 \frac{-\Omega_K H_K^N}{2\Delta} & \Omega_K J_K^N & \frac{\Omega_K H_K^N}{2\Delta}
 \end{array} \right]
 \begin{array}{c}
 \left[ \begin{array}{c}
 Z_0^N \\
 Z_1^N \\
 Z_2^N \\
 \circ \\
 \circ \\
 Z_K^N \\
 Z_{K+1}^N
 \end{array} \right]
 =
 \begin{array}{c}
 \left[ \begin{array}{c}
 l_1 - \Omega_1 \bar{f}_1^N \\
 \bar{g}_1^N \\
 \bar{g}_2^N \\
 \circ \\
 \circ \\
 \bar{g}_K^N \\
 l_K - \Omega_K \bar{f}_K^N
 \end{array} \right]
 \end{array}
 \end{array}$$

$\begin{array}{ccc} K & Z & Y \end{array}$

FIGURE 3      SYSTEM STIFFNESS  
MATRIX, LOAD AND SOLUTION VECTORS



in Ref. 2. The system is tridiagonal in the matrix sense and Potters' form of Gaussian elimination is used to solve for the  $z_{i,j}$ . For the forward pass

$$x_{i,j} = DEE_i \bar{g}_{i,j} - DST_i x_{i-1,j} .$$

For the backward pass

$$z_{i,j} = -P_i z_{i+1,j} + x_{i,j} .$$

The  $4 \times 4$  matrices  $P_i$ ,  $DEE_i$  and  $DST_i$  are independent of the load and solution and are determined only once. They are given by

$$DEE_i = (\bar{B}_i - C_i P_{i-1})^{-1}$$

$$P_i = DEE_i A_i$$

$$DST_i = DEE_i C_i$$

The initial value of  $x_{i,j}$  on the forward pass and the initial value of  $z_{k+1,j}$  on the backward pass are specially determined according to the boundary conditions. Special procedures are also invoked in the case of a shell with an initial and/or final pole [Ref. 2].



### C. STATIC ANALYSIS

Figure 4 illustrates a typical load history and solution path for a static analysis. A solution to a monotonically increasing load is obtained at each load step in an iterative process. The load is incremented by DELOAD each step. A solution is obtained for each Fourier harmonic index considered. The pseudo loads are recomputed each iteration at each load step. The solution at each iteration is tested for convergence. There is no solution to estimate the pseudo loads on the first load step and the result after the first iteration is a linear solution. If convergence occurs, the load is incremented by the amount DELOAD, an estimate of the solution is made by a linear extrapolation of the last two solutions, and the iteration process is repeated.

If convergence is not obtained in a specified number of iterations, ITRMAX, the total load, ALOAD, is decremented by one DELOAD. DELOAD is reduced to 20% of its previous value and becomes the new incremental load. The iteration process is then repeated. The program terminates when DELOAD has been reduced to 20% of its previous value a specified number of times (LCHMAX), or when a specified number of load steps has been taken.

Because of the pseudo load approach, failure to converge to a solution may be due to either of two types of nonlinear behavior illustrated in Figure 4. Curve A experiences a





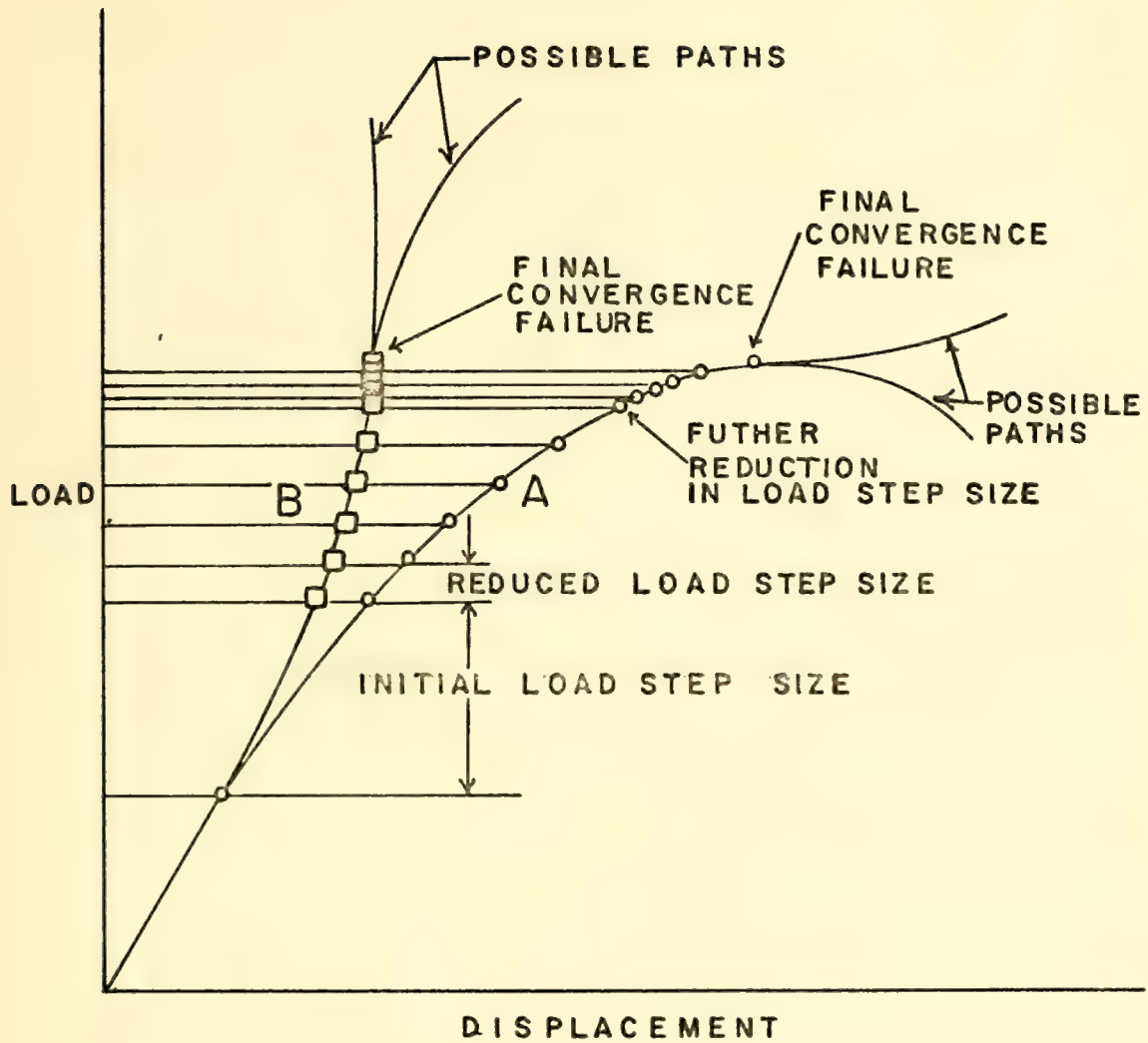


FIGURE 4 NONLINEAR BEHAVIOR  
AFFECTING CONVERGENCE



maximum or inflection point, typical of a softening non-linearity, causing convergence failure. Curve B experiences an ever increasing slope, typical of a stiffening nonlinearity, causing convergence failure.

#### D. DYNAMIC ANALYSIS

The applied load in the dynamic analysis is a function of the time step  $j$ . DELOAD is the incremental time  $\delta t$ . Initial conditions on  $z$  and  $\partial z / \partial t$  are supplied in a user prepared subroutine, INITL. The procedure is essentially the same as the static analysis except the load step becomes the time step. The program terminates under one of two conditions. Either convergence is not obtained in the specified number of iterations, ITRMAX, or the maximum number of time increments, LSMAX, has been reached.



#### IV. DEVELOPMENT OF SATANS-III

##### A. FREE VIBRATION EQUATIONS

Recall the set of four second order differential equations written in Equation (1) was

$$E z'' + F z' + G z = \bar{e} + \bar{\mu} \frac{\partial^2 z}{\partial t^2} \quad (1)$$

In the free vibration analysis the nonlinear terms are ignored and no loading exists. Hence the vector  $\bar{e}$  does not exist under these conditions. Assuming harmonic motion, the solution vector  $z(\xi, T)$  takes the form  $e^{i\omega t T_0} z(\xi)$  where  $T_0$  is a time constant and  $\omega$  is the natural frequency in radians per second. Therefore Equation (1) simplifies to

$$E z'' + F z' + G z = -\omega^2 T_0^2 \bar{\mu} z \quad (5)$$

Using the same spatial finite differencing schemes used in SATANS-I, Equation (5) becomes

$$A_1 z_{i+1} + \bar{B}_1 z_i + C_1 z_{i-1} = -\omega^2 T_0^2 \bar{\mu}_1 2\Delta z_i \quad (6)$$

for  $i = 1, 2, \dots, KMAX$ . The elements of the matrices  $A$ ,  $\bar{B}$ , and  $C$  are the same as those for a static analysis.

Equation (6) can be written in the matrix form of a standard



eigenvalue problem

$$K Z = -\omega^2 T_o^2 M Z \quad (7)$$

where the K matrix is the same as the one shown in Figure 3. The mass matrix M will be described in the following section.

## B. CHARACTERISTICS OF THE PROBLEM

There are three main areas to consider when incorporating a free vibration analysis capability in an existing program. Consideration must be given to the form of the desired results and the expected benefits of the vibration analysis. Consideration must also be given to both compatibility with the existing code and the nature of the problem itself.

### 1. Desired Results

Generally, only the lower frequencies in each Fourier harmonic are sought. The lowest 10 is a common figure. For example, Kalnins, in Reference 6, points out that there is a practical limit to which a linear theory correctly approximates the higher frequencies of a real shell. The mode shapes are also usually desired. In certain applications only a specific range of frequencies is of interest. Some assurance of finding all the frequencies in that range is desired.

### 2. Nature of the Problem

The mass matrix M is diagonal and singular and is of the form





$$M = 2\Delta$$

$$\begin{bmatrix} 0 & & & & & & & \\ & \bar{\mu}_1 & & & & & & \\ & & \cdot & & & & & \\ & & & \cdot & & & & \\ & & & & \bar{\mu}_{i-1} & & & \\ & & & & & \bar{\mu}_i & & \\ & & & & & & \bar{\mu}_{i+1} & \\ & & & & & & & \cdot \\ & & & & & & & \cdot \\ & & & & & & & \bar{\mu}_{KMAX} \\ & & & & & & & & 0 \end{bmatrix}$$

When the meridional, circumferential and radial translational inertial forces are all considered

$$\bar{\mu}_i = \mu_i \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

If only the translational inertial forces in the radial direction are considered

$$\bar{\mu}_i = \mu_i \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$



The stiffness matrix,  $K$ , Figure 3, is unsymmetric and singular in some cases due to the boundary conditions.  $K$  has a bandwidth of 15, or 3 if considered in the matrix sense. When the number of meridional stations is the maximum of 200,  $K$  has an order of 808. If  $K$  were stored in the banded form taking no account of the zeroes in the band, 48 K-bytes of IBM 360/67 main core are required. Potters' elimination scheme requires only 38.4 K-bytes of main core. Because  $K$  is unsymmetric, there is no absolute guarantee, from a mathematical viewpoint, that the eigenvalues and eigenvectors are real. Some capability in this area is sought, both as a trouble shooting aid and as a complex root solver.

Ill-conditioning in the solution may result from several causes. Small separations between the eigenvalues, numerical conditioning of the  $K$  matrix and the loss of significance are some reasons that make eigenvalue/eigenvector determination difficult. Thus it is desirable to have a capability to handle such cases, but not to have to pay the price of determining every eigenvalue and eigenvector of the total system.

### C. SOLUTION TO THE EIGENVALUE PROBLEM

The method of solution to the eigenvalue problem selected for inclusion in SATANS-I is inverse iteration with spectral shifts. This scheme melds with the existing code extremely well. In addition, a capability to handle special ill-



conditioning problems and complex roots has been included. A brief discussion of other possible solutions to the real unsymmetric eigenvalue problem appears in Appendix A.

The inverse iteration procedure is eminently suited for accurate determination of eigenvalues and eigenvectors [Refs. 7 and 8]. The amount of work involved is directly proportional to the number and accuracy of the eigenvalues and eigenvectors sought. The coding is simple and requires a very small increase in core. Lastly, the eigenvectors and eigenvalues are determined simultaneously. The basic scheme is well known [Refs. 7-14]. However, the scheme used in SATANS-III is slightly different and is explained below.

The basic iterative procedure is obtained by rewriting Equation (7) in the form

$$[K - \beta M]W^{r+1} = M Z^r$$

where

$$W^{r+1} = \alpha^{r+1} Z^{r+1}$$

and

$$\alpha = \frac{-1}{(\omega^2 T_o^2) + \beta}$$

where  $r$  denotes the  $r$ th iteration and  $\beta$  is the spectral shift value.





a. The vector  $Z^r$  is assumed for the first iteration and afterward is computed using  $W^r$ . Prior to solving for  $W^{r+1}$  the mass matrix  $M$  must be premultiplied through the vector  $Z^r$ . This consists merely of multiplying the diagonal of  $2\Delta\bar{\mu}_1$  through each  $z_1^r$ . Since the mass may vary along the meridian removal of this variable from the right hand side of the equation was desirable. This was accomplished by dividing each element of the  $A_1$ ,  $\bar{B}_1$  and  $C_1$  matrices by  $2\Delta\mu_1$  at the time of their computation. These additional computations are only performed in the vibration analysis state of the program. The net result of this procedure is that the diagonal of  $\bar{\mu}_1$  merely consists of ones and zeroes.

b.  $W^{r+1}$  is solved for by Potters' elimination scheme as in the original SATANS-I program.

c.  $\alpha^{r+1}$  is determined from the "Rayleigh quotient type" computation

$$\alpha^{r+1} = \frac{(Z^r)^T W^{r+1}}{(Z^r)^T Z^r} \quad (8)$$

which may be thought of as resulting from

$$\frac{(Z^r)^T ([K - \beta M]^{-1} M Z^r)}{(Z^r)^T Z^r}$$

realizing, of course, that this is not the manner in which the computations are performed. In the limit, as  $r$  approaches infinity,  $Z^{r+1}$  approaches  $Z^r$ , when  $Z^r$  is real. This



computation also determines the sign of  $\alpha^{r+1}$ . The vector  $z^{r+1}$  is scaled such that the largest element before each iteration is unity. It sometimes happens in the first few iterations that  $\alpha^{r+1}$  is smaller than the largest element of the vector, and scaling by the largest element prevents vector element build-up. In the limit, as  $r$  approaches infinity,  $\alpha^{r+1}$  becomes equal to the largest element. Hence, if the convergence test is failed, steps a through c are repeated with

$$z^{r+1} = \frac{w^{r+1}}{\max (w_i^{r+1})}$$

d. The primary convergence test is performed on  $\alpha^{r+1}$ . The test is

$$\left| \frac{|\alpha^{r+1}| - |\alpha^r|}{\alpha^{r+1}} \right| = < \text{EPSVIB.}$$

The absolute value of  $\alpha$  is used because due to the spectral shifts  $\alpha$  may be negative. In that case  $\alpha$  changes sign each iteration but converges in absolute value. A secondary convergence test is optional and is performed on the vector elements themselves. The vector test is more stringent than the primary convergence test and increases the computation time. The vector test consists of comparing the difference between two successive iterations with the maximum values of  $u$ ,  $v$ ,  $w$  and  $m_s$ , times EPSVEC, a convergence criterion, for the last iteration.



Recall that

$$Z = \begin{Bmatrix} \cdot \\ \cdot \\ \cdot \\ z_{i-1} \\ z_i \\ z_{i+1} \\ \cdot \\ \cdot \\ \cdot \end{Bmatrix} \quad \text{where } i \text{ denotes the station,}$$

and

$$z_i = \begin{Bmatrix} u_i \\ v_i \\ w_i \\ m_{s_i} \end{Bmatrix} = z_{i,k} \quad \text{where } k = 1(1)4$$

such that  $z_{i,1} = u$  ,  $z_{i,2} = v_i$  ... etc. for each station  $i$ . The vector test may then be stated as follows

$$z_{i,k}^{r+1} - z_{i,k}^r = < \max(z_{i,k}^{r+1}) \times \text{EPSVEC}$$

for  $i = 1(1)\text{KMAX2}$

$k = 1(1)4$ .

EPSVEC = EPSVIB unless otherwise specified on input.

e. When convergence has occurred one more iteration is performed and the vector is scaled using the value of  $\alpha^{r+1}$  given by Equation (8). This yields an indication



of the quality of the solution by inspecting the largest element of the vector, which should be essentially 1.0.

f. The eigenvalue is obtained by taking the reciprocal of  $\alpha^{r+1}$  and subtracting the spectral shift value,  $\beta$ .

$$(\omega^2 T_o^2)^{r+1} = \frac{-1}{(\alpha^{r+1})} - \beta$$

In the program the FORTRAN variables are

```
VREAL = (ONEDO/SCALEW) + ASHIFT
AREAL = SNGL(VREAL)
```

where AREAL and ASHIFT are single precision variables and VREAL, ONEDO and SCALEW are double precision variables.

g. If the program is unable to determine a real eigenvalue in the specified number of iterations (ITRMAX), control is passed to a backup routine, ITCPLX. This routine, which is described in Appendix C, Part Three, was developed to handle special ill-conditioning and non-convergence problems. The results of the routine are the two closest frequencies, real or complex, to the shift point. In addition, diagnostic messages are provided to aid in solving the problem at hand. These additional computations are only performed when the primary iteration routine fails to converge to a solution.

Some of the advantages of the selected scheme have been mentioned in the preceding paragraphs. Another is that iteration provides the ability to determine the frequencies





and mode shapes to almost any precision desired. The iterated vector is computed in double precision in order to handle cases of ill-conditioning and weakly determined frequencies, [Ref. 7]. This adds only slightly to the computation time and has paid off in the ease with which difficult frequencies are determined. Because the mode shapes are also determined, they require no additional calculations and yield an immediate answer to the question of which mode has actually been determined. The answer to this question is not always known in other methods [Ref. 15].

The iteration method avoids all manner of mathematical and numerical problems by actually performing the calculations of the solution procedure. The fact that the K matrix is unsymmetric and that both the K and M matrices are singular has no special impact on the solution procedure. There is no need to reformulate the problem or the program code, a major consideration in the present situation.

#### D. TESTING OF SATANS-III

The program has undergone considerable testing. Because of the size of SATANS-III, the inverse iteration scheme was tested in prototype form first. This included quite a bit of testing of the backup capability planned into the program in addition to the primary scheme [Refs. 7,8,16 and 17]. Appendix B tabulates six test cases for SATANS-III and compares them with published results. The agreement is excellent.



The test cases consist of determining the minimum frequencies of the lower circumferential harmonic (wave) numbers of three cones, a regular right circular cylindrical shell and a shell of positive gaussian curvature. There is also a determination of the lower spectrum of frequencies in each circumferential wave number for the right circular cylindrical shell. Lastly, there is an analysis of the effect of a linearly varying thickness on the minimum frequencies of the lower circumferential wave numbers of the right circular cylindrical shell.



## V. CONCLUSIONS

Application of SATANS-III to many test cases has demonstrated that the inverse iteration technique with spectral shifts is a rapid, accurate and effective method for determining the natural modes and frequencies of the free vibration of shells of revolution. The simultaneous determination of the modes and frequencies provides an opportunity for determining if a mode has been missed, and the supplied shift point operation provides an easy method of determining the missed mode. The fact that sweeping of the vectors was not used in determining the higher frequency modes posed no special problem. In addition, not using vector sweeping allows the use of fewer mass points and more rapid calculation of some frequencies and modes. Dividing the variable mass into the component matrices of the K matrix proved to be an effective method of eliminating this variable from the right side of the vibration equations.

A study of the effect of a linearly varying thickness on the minimum frequencies of a circular cylindrical shell showed that as the percentage thickness of the lower edge over the upper edge increases the cylinder begins to exhibit behavior somewhat similar to that of a cone. The minimum frequency begins to appear in lower circumferential harmonics and the axial half wave number begins to rise.



The program experiences some difficulty with extremely thin shells, corners and very small radii of curvature. This might be alleviated by using a finer grid (more grid points) or as a further modification the incorporation of a variable grid capability.

Another further modification needed is the addition of preloads to the shell. Many of the methods used to determine the modes and frequencies of preloaded shells require accurate initial estimates of the solution to start the procedure. SATANS-III provides these estimates. It is felt that a preload capability can be incorporated into SATANS-III and still keep the program an in-core routine requiring only two core boxes.





## APPENDIX A

### OTHER SOLUTION METHODS

There are primarily two approaches to the unsymmetric matrix eigenvalue problem. The transformation methods are characterized by the operations, finite or infinite in number, being performed on the matrix.\* The iterative methods are characterized by the operations being performed on the solution vector rather than the matrix.

There are basically four transformation methods capable of handling a real, unsymmetric matrix eigenvalue problem. Lanczos' method, the QR and the LR algorithms, are well known [Refs. 7-12]. The QZ algorithm recently developed at Stanford University [Ref. 18] is based on the QR algorithm and can handle a problem in the form  $Ax = \lambda Bx$  with A and B unsymmetric and complex. They all suffer, with respect to the present problem, from the same deficiencies. For example, the greatest portion of the computation time is spent before the first eigenvalue is extracted and they require that the order and size of the matrix be the same. For the case of an unsymmetric banded matrix enough core storage must be set aside for at least half the matrix.

---

\*The general problem  $Ax = \lambda Bx$  may also be thought of as  $Kz = \lambda z$ . In the present problem, the full matrix form of K would require 2,560 K-bytes of core storage on an IBM 360/37.



Lanzcos' method for the unsymmetric case requires core storage for the generation of two sets of biorthogonal vectors in addition to the matrix itself. It is believed that this method would require the greatest storage and computation time. The LR algorithm without interchanges maintains the banded form but suffers from excessive computation time for closely spaced eigenvalues. In addition, the LR algorithm may be numerically unstable in some cases.

By far the most popular and successful transformation method in general use is the QR (and now also the QZ) algorithm. It is stable, fast, accurate and can handle complex numbers. The IMSL group [Ref. 19] is dedicated to speed and accuracy, and virtually all of their routines are variations of the QR and QZ algorithms. The QR algorithm without reduction to Hessenberg form requires a comparatively large number of calculations. Consequently the matrices are generally reduced to Hessenberg form which considerably reduces the computation time required to reach convergence. Again the main objections with respect to the problem considered here is computation time or core storage. Wilkinson [Ref. 7] states "If the LR algorithm with interchanges (which is stable) or the QR algorithm is used the band form above the main diagonal is gradually destroyed." This is also true of any method that employs a transformation to Hessenberg form. See Figure 5.

Another disadvantage of the transformation methods is the necessity of determining the eigenvectors in a separate operation from the natural frequencies.



# REDUCTION OF A MATRIX TO HESSENBERG FORM

## GIVENS' METHOD

### INPUT MATRIX

3.000	2.000	1.000	0.0	0.0	0.0	0.0
4.000	3.000	2.000	1.000	0.0	0.0	0.0
5.000	4.000	3.000	2.000	1.000	0.0	0.0
0.0	5.000	4.000	3.000	2.000	1.000	0.0
0.0	0.0	5.000	4.000	3.000	2.000	1.000
0.0	0.0	0.0	5.000	4.000	3.000	2.000
0.0	0.0	0.0	0.0	5.000	4.000	3.000

### MAJOR STEP NUMBER 1

3.000	2.030	-0.937	0.0	0.0	0.0	0.0
6.403	5.927	-1.659	0.781	0.0	0.0	0.0
0.0	0.341	-0.073	0.625	0.0	0.0	0.0
0.0	6.247	-1.406	2.000	1.000	0.0	0.0
0.0	3.504	-3.123	3.000	2.000	1.000	0.0
0.0	0.0	0.0	4.000	3.000	2.000	1.000
0.0	0.0	0.0	5.000	4.000	3.000	2.000
0.0	0.0	0.0	0.0	5.000	4.000	3.000

### MAJOR STEP NUMBER 2

3.000	2.030	-0.043	0.027	0.0	0.0	0.0
6.403	5.927	-2.189	-0.445	0.0	0.0	0.0
0.0	7.375	5.740	-0.321	1.906	0.529	0.0
0.0	0.0	-0.537	-0.271	0.055	0.0	0.0
0.0	0.0	-2.455	-0.201	1.168	0.848	0.0
0.0	0.0	6.353	0.750	3.000	2.000	1.000
0.0	0.0	2.647	4.242	4.000	3.000	2.000
0.0	0.0	0.0	0.0	5.000	4.000	3.000

FIGURE 5



MAJOR STEP NUMBER 3

3.000	2.030	-0.043	-0.059	-0.920	0.161	0.023	0.0
6.403	5.927	-2.189	-0.279	-1.639	0.757	0.108	0.0
0.0	7.375	5.740	-1.967	-0.137	0.367	-0.194	0.0
0.0	0.0	7.327	5.790	0.369	-0.605	-0.084	1.590
0.0	0.0	0.0	-0.296	-0.607	-0.200	-0.080	0.368
0.0	0.0	0.0	-0.181	-3.050	-0.602	0.033	1.529
0.0	0.0	0.0	-4.163	-1.115	-2.320	0.548	3.000
0.0	0.0	0.0	5.781	0.0	-1.839	2.050	

MAJOR STEP NUMBER 4

3.000	2.030	-0.043	-0.059	0.047	-0.619	5.698	-0.066
6.403	5.927	-2.189	-0.279	0.112	-1.503	-0.988	-0.155
0.0	7.375	5.740	-1.967	-0.128	-0.242	-0.088	0.178
0.0	0.0	7.327	5.790	0.369	-0.709	0.033	0.312
0.0	0.0	0.0	-0.296	-0.607	-0.397	-0.233	0.063
0.0	0.0	0.0	-0.181	-3.050	-1.720	-1.611	-0.018
0.0	0.0	0.0	-4.163	-1.115	-1.108	-1.250	-0.148
0.0	0.0	0.0	5.781	0.0	-2.127		

MAJOR STEP NUMBER 5

3.000	2.030	-0.043	-0.059	0.047	-0.066	0.930	-0.019
6.403	5.927	-2.189	-0.279	0.112	-0.108	1.747	-0.441
0.0	7.375	5.740	-1.967	-0.128	0.356	-0.049	-0.214
0.0	0.0	7.327	5.790	0.369	-0.208	-0.547	-0.413
0.0	0.0	0.0	-0.296	-0.607	-0.333	-0.441	0.098
0.0	0.0	0.0	-0.181	-3.050	-1.237	1.595	-2.040
0.0	0.0	0.0	-4.163	-1.115	0.227	0.034	-1.639
0.0	0.0	0.0	5.781	0.0	-0.936	-2.757	

MAJOR STEP NUMBER 6

3.000	2.030	-0.043	-0.059	0.047	-0.066	0.200	-0.909
6.403	5.927	-2.189	-0.279	0.112	-0.108	-0.018	-1.802
0.0	7.375	5.740	-1.967	-0.128	0.356	-0.220	-0.003
0.0	0.0	7.327	5.790	0.369	-0.208	0.273	0.629
0.0	0.0	0.0	-0.296	-0.607	-0.333	0.165	-0.414
0.0	0.0	0.0	-0.181	-3.050	-1.237	-1.693	-2.044
0.0	0.0	0.0	-4.163	-1.115	0.227	-2.270	-2.099
0.0	0.0	0.0	5.781	0.0	-0.936	-0.156	0.664

FIGURE 5 CONTINUED









An approach taken by some [Ref. 20] is to cast the problem into a form that is easily handled by one of the more popular methods. The difficulty here lies in that many of the operations depend on at least one of the matrices in the general problem  $Ax = \lambda Bx$  being nonsingular in all circumstances. This was not the case with the present problem. It was felt that the amount of work involved in either reformulating the problem or reducing the matrices to a form that could be transformed easily involved too much computational effort.

A method classed as iterative, but one that does not operate on the vector, is that of evaluating a determinant. This has been applied to the present problem by other researchers [Refs. 6, 21, 22]. One of the difficulties with this method is that one or two initial guesses at the eigenvalue must be made to start the process. The better the guess the faster the convergence. Another is that, unlike the inverse iteration process, there is no inherent determination of the eigenvalue spectrum under consideration. Having no a priori knowledge of the solution to a particular problem it is possible to miss a mode and not realize it [Ref. 15]. This method also requires separate determination of the eigenvectors.

When the inverse iteration procedure was chosen as the solution procedure it was also decided that orthogonality and sweeping of the vectors would not be employed. Because the K matrix is never determined explicitly, the use of a



sweeping matrix was not considered. Sweeping, if it would be employed, would have to operate solely on the vectors [Ref. 23]. Because the K matrix is unsymmetric the right hand vectors (mode shapes) are not orthogonal through K or M, although they are independent. The vectors that are orthogonal are the right and left hand vectors. If modal sweeping is used a total of 6 K-bytes of main core is required for each vector swept (3 K-bytes per vector; two vectors per mode). An increase in computation time also results from the sweeping operation.

Orthogonality relationships for the mode shapes can be developed in the continuous theory, but if an unsymmetric K matrix results from the implementation of the theory then the right hand vectors are not, in fact, orthogonal. In some situations the right hand vectors have been treated as if they were orthogonal when they are not [Refs. 13, 24]. The critical factor, in this case, is the number of mass points chosen to approximate the continuous system. As the number of mass points increases the approximation to the correct mode shapes (which are orthogonal) becomes more accurate. For this condition the sweeping operation becomes effective, but the computation time also increases. With a few mass points the eigenvalues may be accurately determined and the computation time reduced considerably. However, the sweeping operation then serves only to introduce unwanted errors. It was felt that to effectively employ sweeping would result in a large computation time and core requirement



that could not be justified by the results. In addition, in some of the cases investigated using sweeping of the right hand vectors as if they were orthogonal, there seemed to be no immediately apparent correlation between an increase in the number of mass points and the resultant increase in the quality of the pseudo orthogonality of the right hand vectors.





# APPENDIX B

## NATURAL FREQUENCIES OF SOME SELECTED SHELLS

TABLE I

MINIMUM NATURAL FREQUENCIES (Hz) OF A  
THIN CIRCULAR CYLINDRICAL SHELL<sub>(1)</sub>

<u>Circumferential Wave Number, n</u>	<u>Ref. 25</u>	<u>Ref. 26</u>	<u>SATANS-III</u>
1		1446.0	1446.2
2	633.5	634.5	635.5
3	326.7	327.5	328.2
4	202.3	203.8	203.7
5	159.9	162.6	160.9
6	168.0	171.6	168.7
7	206.0	210.1	206.5
8	261.0	265.4	261.6
9		331.9	328.0
10	403.4	408.0	404.1
11		492.8	488.8
12	581.1	585.5	582.0
13		687.3	683.4
14	791.8	796.9	793.0
15		914.7	911.0
16		1041.0	1036.9
17		1175.0	1171.1
18		1317.0	1313.5
19		1467.0	1463.9
20		1626.0	1622.5

(1) Configuration 1, Table VI.



TABLE II  
NATURAL FREQUENCIES (Hz) OF A THIN CIRCULAR CYLINDRICAL SHELL<sup>(1)</sup>

Circum. Wave Number, n	Axial Half-Wave Number, m											
	m = 1			m = 2			m = 3			m = 4		
	A	B	C	A	B	C	A	B	C	A	B	C
4	202.3	203.8	203.7	696.1	697.5	698.0						
5	159.9	162.6	160.9	483.0	484.6	484.6	960.6	962.5	963.3			
6	168.0	171.6	168.7	370.5	372.7	371.9	724.0	725.9	726.5			
7	206.0	210.1	206.5	325.1	328.1	326.3	580.8	583.2	583.2			
8	261.0	265.4	261.6	329.2	332.9	330.2	506.1	509.0	508.3	768.3	771.0	771.2
10	403.4	408.0	404.1	429.2	433.7	430.1	506.6	510.7	508.4	649.9	653.6	653.1
12	581.1	585.9	582.0	594.9	599.7	596.0	632.3	637.1	633.9	706.4	711.1	709.0
14	791.8	796.9	793.0	801.8	806.9	803.0	824.7	829.9	826.2	867.6	872.7	869.7

(1) Configuration 1, Table VI.  
(A) Reference 25  
(B) Reference 26  
(C) SATANS-III



TABLE III

MINIMUM NATURAL FREQUENCIES (Hz) OF A FREELY  
SUPPORTED CIRCULAR CYLINDRICAL SHELL<sub>(1)</sub>

Circum. Wave Number, n	With Inplane Inertia DIAG = (1,1,1,0)		Without Inplane Inertia DIAG = (0,0,1,0)	
	f	$\Xi$ (2)	f	$\Xi$
1	125.613	0.372	168.177	0.498
2	61.26	.182	69.24	.285
3	33.11	.0981	35.09	.104
4	19.92	.0590	20.58	.0610
5	13.62	.0403	13.90	.0412
6	10.07	.0298	10.21	.0303
7	8.494	.0252	8.583	.0254
8	8.285	.0245	8.351	.0247
9	9.057	.0268	9.114	.0270
10	10.33	.0306	10.38	.0308
11	12.08	.0358	12.13	.0359
12	14.24	.0422	14.29	.0423
13	16.58	.0491	16.63	.0493
14	19.16	.0568	19.21	.0569
15	21.96	.0651	22.01	.0652
16	24.94	.0739	24.99	.0740
18	31.56	.0935	31.61	.0936
20	38.95	.115	39.00	.115
22	47.14	.140	47.19	.140
24	56.12	.166	56.10	.166
26	65.87	.195	65.91	.195
28	76.40	.226	76.41	.226

(1) Configuration 2, Table VI.

(2) Dimensionless Frequency Parameter  $\Xi = \omega \bar{r} \sqrt{\frac{m(1-\nu^2)}{E_y}}$



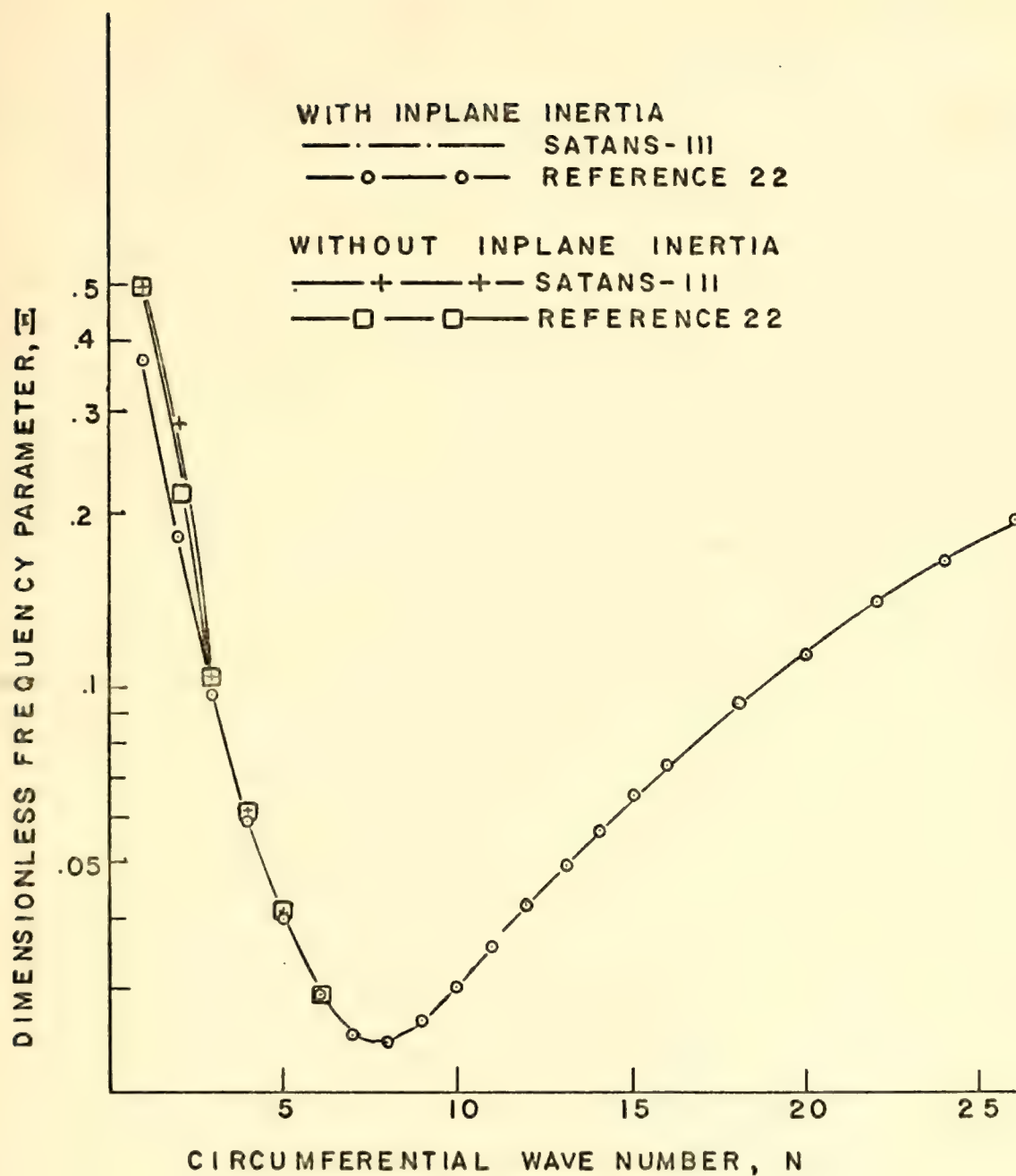


FIGURE 6 MINIMUM NATURAL FREQUENCIES OF A FREELY SUPPORTED CIRCULAR CYLINDRICAL SHELL CONFIGURATION 2, TABLE VI





TABLE IV  
 MINIMUM NATURAL FREQUENCIES OF A  
 SHELL OF POSITIVE GAUSSIAN CURVATURE<sub>(1)</sub>

Dimensionless Frequency Parameter

$$\Xi = \omega \bar{r} \sqrt{\frac{m(1-\nu^2)}{E_y}}$$

<u>Circumferential Wave Number, n</u>	<u>Ref. 27</u>	<u>Ref. 22</u>	<u>SATANS-III</u>
1	0.411	0.412	0.411
2	.360	.362	.360
3	.340	.340	.340
4	.331	.331	.331
5	.327	.327	.327
6	.324	.324	.324
7	.323	.322	.323
8	.322	.321	.322
9	.321	.321	.321
10	.321	.321	.321

(1) Configuration 3, Table VI.



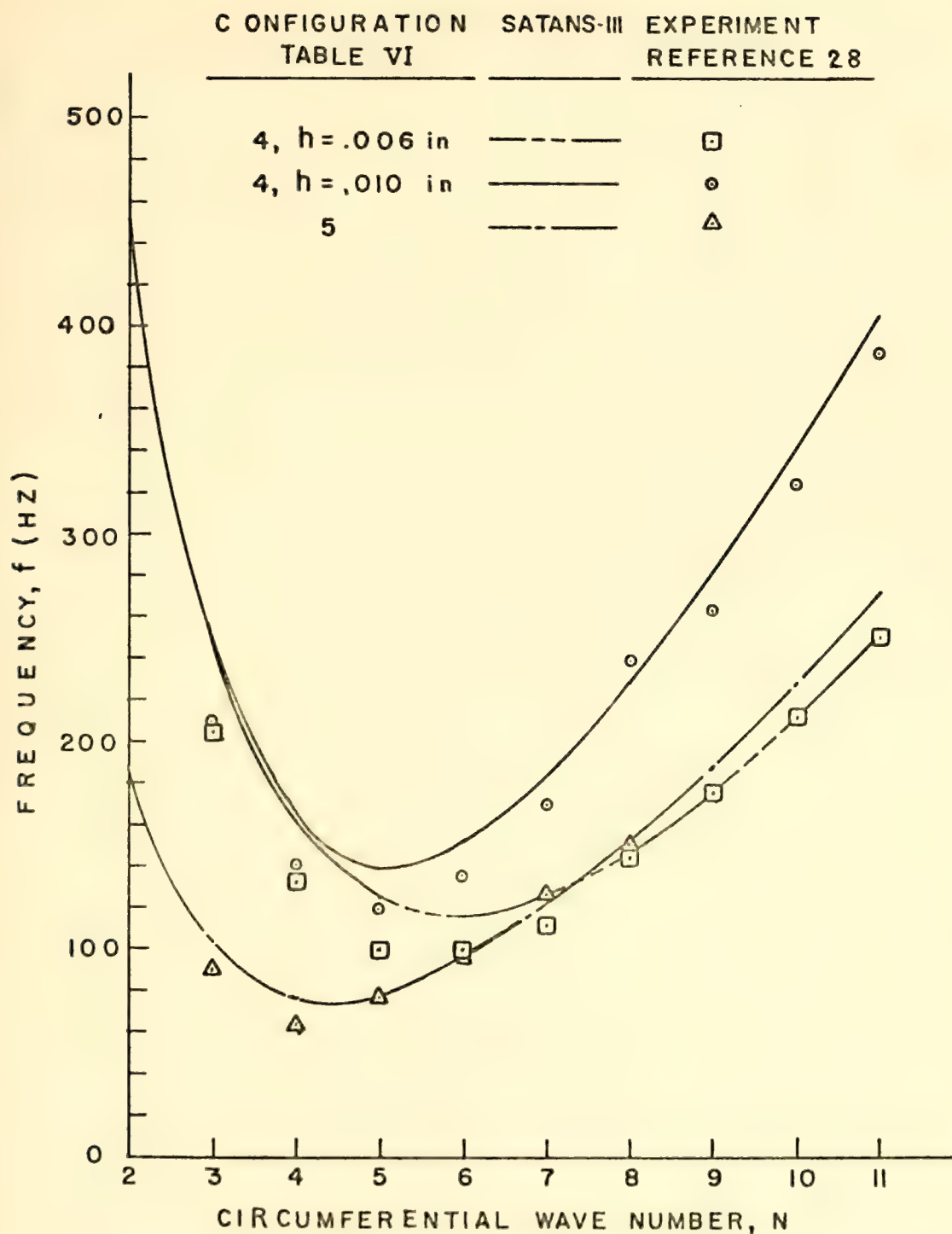


FIGURE 7

MINIMUM NATURAL FREQUENCIES OF 3 CONES



TABLE V

EFFECT OF A LINEARLY VARYING THICKNESS ON MINIMUM FREQUENCIES (Hz)<sub>(1)</sub>

$$h = .1(x - 1.)(s/S_T) + .1$$

Circum. Wave Number, n	x = 1		x = 2		x = 4		x = 7		x = 10	
	f	m(2)	f	m	f	m	f	m	f	m
1	125.6	1	122.1	1	112.7	1	104.4	1	99.01	1
2	61.26	1	60.72	1	58.44	1	56.09	1	54.28	1
3	33.11	1	32.94	1	32.12	1	31.30	1	30.71	1
4	19.92	1	20.19	1	20.02	1	20.26	1	20.81	1
5	13.62	1	13.85	1	14.80	1	16.43	1	18.92	1
6	10.07	1	10.87	1	13.03	1	17.22	1	21.63	1
7	8.494	1	10.14	1	14.08	1	20.13	1	25.43	1
8	8.285	1	10.91	1	16.37	1	23.25	1	28.87	2
9	9.057	1	12.60	1	18.93	1	26.20	2	32.27	2
10	10.33	1	14.77	1	21.50	2	29.25	2	35.77	2
12	14.24	1	19.65	1	27.05	2	35.75	2	43.11	3
14	19.16	1	25.22	2	33.31	3	42.88	3	51.01	3
16	24.94	1	31.57	2	40.31	3	50.71	4	59.57	4
18	31.56	1	38.73	3	48.10	4	59.28	4	68.84	4
20	38.95	1	46.70	3	56.69	4	68.64	4	78.87	5

(1) Configuration 6, TABLE VI

(2) Axial Half-wave Number



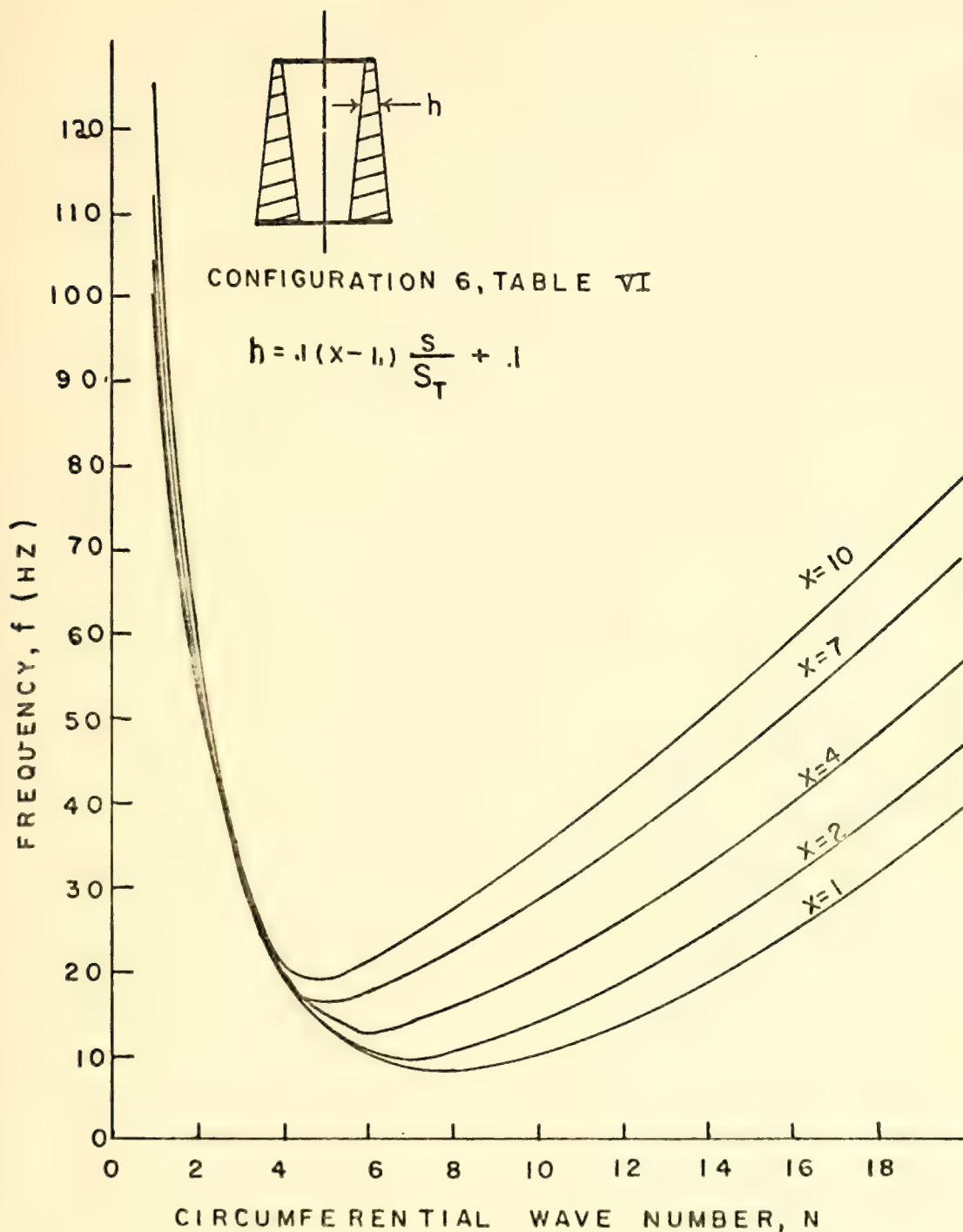


FIGURE 8 EFFECT OF A LINEARLY VARYING THICKNESS ON MINIMUM FREQUENCIES





TABLE VI  
SHELL CONFIGURATIONS

Configuration	1	2	3
r (in.)	6.0	100.	$A = .5 - s/3$ $\cos(A) = 1.879$
$\bar{r}$ (in.)	6.0	100.	1.
$r_1$ (in.)	6.0	100.	.754
$r_2$ (in.)	6.0	100.	.754
$R_s$ (in.)	$\infty$	$\infty$	3.
$R_\theta$ (in.)	6.0	100.	$r/\cos(A)$
h (in.)	.015	.1	.001
$h_1$ (in.)	.015	.1	.001
$h_2$ (in.)	.015	.1	.001
ht (in.)	24.0	300.	2.876
$S_T$ (in.)	24.0	300.	3.
$E_y (10^8 \text{ psi})$	.3	.3	$10^{-8}$
$\nu$	.3	.3	.3
$m(10^{-3} \text{ lb-sec}^2/\text{in}^4)$	.732	.733	$10^3$
BNDRY. COND. (1)			
INITIAL EDGE	FS	FS	FS
FINAL EDGE	FS	FS	FS

(1) Table VII



TABLE VI (Continued)

Configuration	4	5	6
$r$ (in.)	$r_1 + .183S$	$r_1 + .274S$	100.0
$\bar{r}$ (in.)			100.0
$r_1$ (in.)	3.35	4.81	100.0
$r_2$ (in.)	5.58	10.9	100.0
$R_s$ (in.)	$\infty$	$\infty$	$\infty$
$R_\theta$ (in.)	1.017r	1.04r	100.0
$h$ (in.)	$\frac{.006}{.010}$	.026	$.1 \cdot (x-1) \cdot (s/S_T) + .1$
$h_1$ (in.)	$\frac{.006}{.010}$	.026	.1
$h_2$ (in.)	$\frac{.006}{.010}$	.026	$.1 \cdot x$
$ht$ (in.)	12.0	21.3	300.
$S_T$ (in.)	12.2	22.15	300.
$E_y$ ( $10^8$ psi)	.28	.148	.3
	.29	.3	.3
$v_m$ ( $10^{-3}$ lb-sec <sup>2</sup> /in. <sup>4</sup> )	.740	.4244	.733
BNDRY. COND <sub>(1)</sub>			
INITIAL EDGE	C	C	FS
FINAL EDGE	F	F	FS

(1) Table VII



TABLE VII  
BOUNDARY CONDITION  
SPECIFICATION<sub>(1)</sub>

Case	Specification
Free (F)	$N_s = \hat{N}_{s\theta} = \hat{Q}_s = M_s = 0$
Freely Supported (FS)	$N_s = M_s = 0$ $V = W = 0$
Simply Supported (SS)	$M_s = 0$ $U = V = W = 0$
CLAMPED (C)	$U = V = W = \phi_s = 0$

(1) The boundary conditions are specified on the total variables  $N_s$ ,  $\hat{N}_{s\theta}$ ,  $\hat{Q}_s$ ,  $\phi_s$ ,  $U$ ,  $V$ ,  $W$  and  $M_s$ , as depicted in Figure 2, for all Fourier harmonics considered.



## APPENDIX C

### CONVERTING SATANS-I TO SATANS-III

#### PART ONE

#### PROCEDURE FOR ALTERING COMMON BLOCKS IN TRANSFORMING SATANS-I TO SATANS-III.

1. Remove vector X(4,200) from BL4 and insert in BL104
2.     "         "         Z(4,220)     "     BL6     "         "         "         "
3.     "         "         ZO(4,220)     "     BL101     "         "         "         "
4.     "         "         Z2(4,220)     "         "         "         "         "
5.     "         "         Z3(4,220)     "         "         "         "         "
6. The order of the vectors in common block BL104 is  
    Z, ZDOT, X, ZO, Z2, Z3.
7. Change the order of the vectors in common block  
    BL24 from DL, DG, DF to DG, DL, DF.
8. The order of the remaining vectors in common blocks  
    BL4, BL6 and BL101 should not be altered.





TABLE VIII  
CONTENTS OF SATANS-III COMMON BLOCKS

COMMON BLOCK LABEL	CONTENTS
IBL1	MNMAX
IBL2	N(10), MNINIT
IBL3	M0, M1, M2, M3
IBL4	KMAX, KL
IBL5	IBCINL, IBCFNL
IBL6	KLL
IBL7	MNMAXO, MAXD(10), MAXS(10), MAXSY(10), IS(10,10), JS(10,10), ID(10,10), or <sup>(1)</sup> JD(10,10), IJS(10) MNMAXO, MAXD(10), MAXS(10), MAXSY (10), MODVIB(3,100), ITEMP(110)
IBL8	LSTEP, ITR
IBL9	MAXM
IBL10	IFREQ, NTHMAX
IBL11	ICORFL, IPASS
IBL12	KMAX1, KMAX2, NCONV
IBL13	ITRMAX, LSMAX
BL1	A(4,4), BEE(4,4), C(4,4)
BL3	PR(10), PX(10), PT(10)
BL4 <sup>(2)</sup>	P(4,4,200), ZF1M(4,4,10), ZF2M(4,4,10), ZF3M(4,4,10), ZF4M(4,4,10)



TABLE VIII (Continued)

BL5	TT(10), MT(10), DT(10), DMT(10)
BL6 <sup>(2)</sup>	SOE, OSE, ALOAD
BL7	D1, S1
BL8	R(200), GAM(200), OMT(200)
BL9	FFS(4,10), ELIS(4), GEES(4,10)
BL10	PHIX(10), PHIT(10), PHI(10)
BL11	OMXI(200), PHEE, T0, T2
BL12	TDLI, TDEL
BL13	OMEG1(4,4), CAPL1(4,4), OMEGL(4,4), CAPLL(4,4), UNIT(4,4)
BL14	LAM2, LSD18, LSD1N
BL15	NU, U1(10), V1(10), W1(10), V2(10), U2(10), W2(10), U3(10), V3(10), W3(10)
BL16	EPS
BL17	DEL
BL18	EL1(4), ELL(4)
BL19	TH(36)
BL20	DEOMX(200)
BL23	JAY(4,4), H(4,4)
BL24 <sup>(1)(2)</sup>	DG(4,4,10), DL(4,4,10), DF(4,4,10)
	or DG(4,4,4), RANGE(100,4), TOP, BOTTOM, DELSHF, TOLSHF, RZ(12)
BL25	E(4,4), F(4,4), G(4,4)
BL27	BX3(10), BT3(10), BXT3(10), BE3(10)



TABLE VIII (Continued)

BL28	EXX3(10), ETT3(10), ETX3(10), EXT3(10), EX3(10), ET3(10)
BL29	BX1(10), BT1(10), BXT1(10), BE1(10), BX2(10), BT2(10), BXT2(10), BE2(10)
BL30	EXX1(10), ETT1(10), ETX1(10), EX1(10), ET1(10), EXX2(10), ETT2(10), ETX2(10), EX2(10), ET2(10), EXT2(10)
BL31	DELSQ, EXT1(10)
BL32	TKN, ELAST, CHAR, SIGO
BL34	DEE(4,4,200), DST(4,4,200)
BL100 <sup>(3)</sup>	TEEO, \$DYNMC
BL101 <sup>(2)</sup>	DELSD
BL102	DELOAD
BL103	MASS(200)
BL104 <sup>(1)(2)</sup>	Z(4,220), ZDOT(4,220), X(4,200), ZO(4,220), Z2(4,220), Z3(4,220) or WV(4,203), XV(4,203), YV(4,203), FILL(28), SHFTPT(3,100)
BL110	TX(10), TTH(10), TXT(10), MX(10), MTH(10), MXT(10), QS(10)
BL111	ABZ, ABZO, ABZN, ABZ3, DD2
BLPLOT	IRADII, IGAMMA, IOMEGS, IOMEGT, IDEOMS, IBSTIF, IDSTIF, IBBSTF, IDDSTF, IPR, IPS, IPT, ITT, IMT, IDTT, IDMT, INS, INTH, INSTH, IQS, IMS, IMTH, IMSTH,



TABLE VIII (Continued)

	IU, IV, IW, IPHIS, IPHIT, IPHI, \$PLOTS, \$MODAL
BLPLT1	XRADII(100), YGAMMA(100), YOMEGS(100), YOMEGT(100), YDEOMS(100), YBSTIF(100), YDSTIF(100), YBBSTF(100), YDDSTF(100), YPR(100), YPS(100), YPT(100), YTT(100), YMT(100), YDTT(100), YDMT(100), YNS(100), YNTH(100), YNSTH(100), YQS(100), YMS(100), YMT(100), YMSTH(100), YU(100), YV(100), YW(100), YPHIS(100), YPHIT(100), YPHI(100), XSTATN(100)
BLDATA	TITLE, NO, IMODE, NDIMEN, IPRINT, LCHMAX, IC
BLRUN	IRNAGN
BLK1 <sup>(4)</sup>	VMASS, NN2, \$VIBES
BLK2	IVIBES, IVB
BLK3	TWOPI, IBEGIN, IEND, IGO, ISTOP
BLK4	ZERODO, PTFIVE, ONEDO, SMALL, VREAL, VIMAG, AREAL
BLK5	\$FLAG, \$TAG, \$DOVEC
BLK6	NJ, NK, I4, IRGOLD, NUMBER, ITWO
BLK7	SCALEX, SCALEW, SCALE3
BLK8	EPSVIB, EPSVEC, EPSAIT
BLK9	ITR1, ITR2





TABLE VIII (Continued)

BLK10	ASHIFT, ASHFUM, ATSHF, BII, TOLUP, TOLDWN, TOLCLS
BLK11	DIAG(4)
BLK12	NTRY
BLK13	DIF1, DIF2, DIF3, DIF4, TOLLOW, TOLHI, TOLRNG

- (1) Identifies a common block of storage that is overlaid by the vibration analysis portion of the program.
- (2) Identifies a common block that has been altered in transforming SATANS-I to SATANS-III.
- (3) The character \$ indicates a LOGICAL \* 1 variable.
- (4) This and subsequent common blocks designated BLKxx are additions required in transforming SATANS-I to SATANS-III.
- (5) Single Precision (REAL\*4) variables: JAY, LAM2, LSD18, LSD1N, MASS, MT, NU
- (6) Double Precision (REAL\*8) variables: EPSVIB, EPSVEC, EPSAIT, ONEDO, PTFIVE, SCALEW, SCALEX, SCALE3, SMALL, TWOPI, VREAL, VIMAG, WV, XV, YV, ZERO DO



TABLE IX  
SOME FORTRAN VARIABLES

AREAL	single precision form of $-(\omega^r T_0)^2 = +1/\alpha^r$
ASHIFT, ASHFUM, ATSHF	spectral shift points, $\beta$
BOTTOM	lower bound on frequency range determined in subroutines RANGER and BTM
CHAR	characteristic shell reference length, $a$
DIAG(4)	diagonal of mass matrix $\bar{\mu}$ . Location WV(J, 203), J = 1(1)4 stores the double precision form of DIAG(4).
ELAST	$E_0$
EPSVIB	eigenvalue convergence criterion
EPSVEC	eigenvector convergence criterion
EPSAIT	Aitkin extrapolation convergence criterion
KEY1	input parameter = MODVIB(1,K) for the Kth vibration analysis
KEY2	input parameter = MODVIB(2,K) for the Kth vibration analysis = N(1)
KEY3	input parameter = MODVIB(3,K) for the Kth vibration analysis; determines \$DOVEC
MODVIB(3,100)	storage array for input parameters KEY1, KEY2, KEY3
RANGE(100, 4)	storage array for frequency range parameters
SCALEW, SCALEX, SCALE3	$\alpha^r$ , $\alpha^{r-1}$ and $\alpha^{r-2}$ respectively
SIGO	$\sigma_0$
SHFTPT(3, 100)	storage array for input values; shift points or frequency range boundaries



TABLE IX (Continued)

TEEO	$T_o$
TKN	$h_o$
TOP	upper bound of frequency range determined in subroutine RANGER and TOPPER
VMASS	$\mu_i$
VREAL	double precision form of $-(\omega^r T_o)^2$  $= \frac{+1}{\alpha^r} = \frac{+1.}{SCALEW}$
VIMAG	imaginary part of a complex eigenvalue
WV(4, 203), XV(4, 203), YV(4, 203)	double precision solution vectors for the r, r-1 and r-2 vibration analysis iterations
WT(812), XT(812), YT(812)	singly dimensioned counterparts of WV, XV AND YV.
Z(4, 220)	single precision solution vector
ZV(4, 200)	double precision working vector
\$DOVEC	logical operating parameter. If KEY3 = 1, then \$DOVEC is true and the vector convergence test will be performed.
\$FLAG	logical operating parameter. If \$FLAG is true, a real eigenvalue is being dealt with. If \$FLAG is false, the program is dealing with a complex eigenvalue.
\$TAG	logical operating parameter. If \$TAG is true, lower portion of frequency range is being searched. If \$TAG is false, upper portion is being searched.
\$VIBES	Logical input parameter to designate a free vibration problem.



## APPENDIX C

### PART TWO

#### MODIFYING SATNAS-I SUBROUTINES

I. Modify common block statements BL4, BL6, BL24, BL101, and BL104, as specified in Part One, in all affected subroutines. In some cases this means replacing one common block statement with two or vice versa. Hereafter all common blocks are referred to as having been modified. The program is written in FORTRAN and all statements referred to are FORTRAN statements.

#### II. SUBROUTINE SATANS

1. After card 060 insert the statement

```
REAL*8 EPSVIB, EPSVEC, EPSAIT
```

2. After card 100 insert common block

```
IBL7.
```

3. After card 290 insert common blocks BLK1, BLK2, BLK8, BLK9, and BLK11.

4. Replace the dimension statement, card 300, with the two statements

```
DIMENSION MODVIB(3,100), SHFTPT(3,100), TITLE(18)  
EQUIVALENCE (MODVIB(1,1), IS(1,1)), (SHFTPT(1,1),  
Z3(1,146))
```

5. Replace cards 350 and 360 with

```
READ(5,101) NO, $DYNMC, $VIBES, IMODE, NDIMEN,  
NTHMAX, IFREQ, IPRINT,
```

```
1 IBCINL, IBCFNL, KMAX, MNMAX, MAXM, LSMAX, LCHMAX, ITRMAX, IC
```





6. Replace card 490 with the following five cards  
IF(.NOT.\$VIBES) GO TO 10  
READ(5,108) IVIBES,DIAG,EPSVEC,ITR2  
READ(5,109) (((MODVIB(J,I),J=1,3),(SHFTPT(K,I),  
K=1,3)), I=1,IVIBES)  
10 READ(5,107) IRNAGN  
IF(\$VIBES) CALL INITAA
7. After card 650 insert the statement WRITE(6,211)
8. After card 710 insert the statement  
IF(\$VIBES) GO TO 11
9. Replace card 750 with the following 7 cards  
GO TO 6  
card 1180  
card 1190  
GO TO 6  
11 WRITE(6,252) KMAX,ITR1,IVIBES,EPS  
WRITE(6,247) CHAR,TKN,ELAST,SIGO,TEEO,NU  
6 IF(.NOT.\$PLOTS) GO TO 9
10. Replace cards 1170 and 1200 with the statement  
9 IF(NTHMAX.EQ.0) GO TO 7
11. Replace card 1340 with  
101 FORMAT(I5,L3,L2,14I5)
12. Insert the following format statements after card  
1400.  
108 FORMAT(I5,4F1.0,F16.8,I5)  
109 FORMAT(3I5,3F15.8)



13. Insert the following format statement after card 2250.

```
252 FORMAT(' ',4X,'NUMBER OF STATIONS-----  
-----',I3,/5X,'M  
1AXIMUM NUMBER OF ITERATIONS-----',  
I3,/5X,'NUMBER OF VIBRATI  
2ON ANALYSES-----',I3,/5X,'CONVERGENCE  
CRITERION(EIGENVALUE  
, , 3S)-----',E12.4//)
```

### III. SUBROUTINE DYNAMC

1. Insert common block BLK1 after card 770.
2. Insert the following statements after card 1870.

```
ICHCK2 = ICHCK2 + ICHCK1  
IF($PLOTS.AND.(ICHCK2.NE.O)) CALL PLOT1(1)  
IF($VIBES) CALL VIBERS(&500)
```

### IV. SUBROUTINE PMATRIX

1. After card 200 insert the statement  
IMPLICIT LOGICAL\*1 (\$)
2. After card 340 insert common blocks BL104, BLK1,  
and BLK10.
3. After card 410 insert the statements  
ASHFUM = ASHIFT  
ATSHF = ASHFUM
4. Place card 830 after card 850.
5. After card 840 insert the statement  
IF(\$VIBES) GO TO 2.



6. Replace card 860 with

2 IJ = 1 + KMAX \* (MN-1)

V. SUBROUTINE EFG

1. Insert common block BLK1 after card 190

2. After card 220 insert the statements

VMASS = MAS

IF(\$VIBES) MAS = 0.0

VI. SUBROUTINE ABC

1. After card 050 insert the statement

IMPLICIT LOGICAL \* 1 (\$)

2. After card 090 insert common blocks BLK1, BLK10,  
BLK11 and the two statements

DIMENSION TEMP(48)

EQUIVALENCE (TEMP(1),A(1,1))

3. Replace card 190, the RETURN statement, with the  
following series of statements.

IF (.NOT.\$VIBES) GO TO 4

VMASS = 1.0/(TDEL \* VMASS)

DO 2 I = 1, 48

2 TEMP(I) = TEMP(I) \* VMASS

DO 3 I = 1,4

BEE(I,I) = BEE(I,I) - ASHFUM\*DIAG(I)

3 CONTINUE

4 RETURN



## VII. SUBROUTINE OUTPUT

1. After card 500 insert common block BL104
2. After card 640 insert common block BLK1
3. After card 670 insert the statement  
    IF(\$VIBES) GO TO 182.
4. After card 890 insert the statement  
    IF(\$VIBES) GO TO 991.
5. Replace card 2320 with  
    IF(.NOT.\$VIBES) WRITE(6,749)N(MN)

## VIII. SUBROUTINE PLOTIT

On cards 080, 090, 100 and 460 change the FORTRAN variable name RANGE to RANGES.

## IX. SUBROUTINE DRAWIT

On cards 1860, 1880, 1970, 1980, 1990 and 2000 change the FORTRAN variable name RANGE to RANGES.





## APPENDIX C

### PART THREE

#### NEW SUBROUTINES - THEORY AND FLOW CHARTS

Following are descriptions of the subroutines that, added to SATANS-I, constitute the major portion of the conversion from SATANS-I to SATANS-III.

##### I. SUBROUTINE VIBERS

Subroutine VIBERS is the primary controller for the vibration analysis portion of the program. After the initial input parameters have been printed, subroutine DYNAMC passes control to subroutine VIBERS. Depending on the vibration input parameters subroutine VIBERS calculates and prints the spectral shiftpoint and passes control to the appropriate subroutines. The flowchart for subroutine VIBERS is depicted in Figure 10.

##### II. SUBROUTINE ITREAL

The purpose of this subroutine is to control the primary iteration process in determining real eigenvalues. The convergence test on the eigenvalues is performed in this subroutine. The vector convergence test is not. That test is performed in subroutine ITRATE. Recall that

$$\alpha^r = \frac{-1}{(\omega^r T_0)^2} = \text{SCALEW}, \quad \text{VREAL} = -(\omega^r T_0)^2$$



and AREAL is the single precision counterpart of VREAL.

The flowchart for subroutine ITREAL is depicted in Figure 11.

### III. SUBROUTINE ITRATE

In subroutine ITRATE the primary iteration calculations are performed. In addition, if it is to be performed, the vector convergence test is performed in this subroutine.

As discussed previously the primary iteration scheme uses Potters' Gaussian elimination method. For the forward pass

$$ZV_{i,k} = DEE_{i,j,k} * WV_{j,k+1} - DST_{i,j,k} * ZV_{j,k-1}$$

$$\text{for } k = 1(1)KMAX2, \quad i = 1(1)4 \quad \text{and} \quad j = 1(1)4.$$

Similarly for the backward pass

$$WV_{i,k+1} = ZV_{i,k} - P_{i,j,k} * WV_{j,k+2}$$

$$\text{for } k = 1(1)KMAX2, \quad i = 1(1)4 \quad \text{and} \quad j = 1(1)4$$

Separate calculations are used to start and finish the calculations as well as handle shells with initial and/or final poles.

WV is the solution vector for the  $r^{\text{th}}$  iteration. XV and YV are, respectively, the solutions for the  $r-1$  and  $r-2$  iterations.



WT, XT, and YT are the singly dimensioned versions of, respectively, WV, XV and YV. The flowchart for subroutine ITRATE is depicted in Figure 12.

#### IV. SUBROUTINE ITCPLX

##### A. CONVERGENCE FAILURE

A few of the conditions that may cause the inverse iteration scheme to fail are:

1. Eigenvalues may be too close together for separation to occur in the specified maximum number of iterations.
2. A complex conjugate pair of eigenvalues may cause convergence failure.
3. If the two lowest eigenvalues are of equal magnitude and opposite sign, convergence is extremely difficult, if not impossible. This may be caused by shifting. If, for example, the shift point is 6 and three eigenvalues are 4, 6 and 8, the resulting lowest two are plus and minus 2.
4. The occurrence of a repeated eigenvalue almost prohibits passing the vector convergence test as do conditions two and three.
5. A poorly conditioned matrix or shifting too close to an eigenvalue may cause a numerically unstable situation in the machine.



The above conditions have not yet occurred when using SATANS-III, but were forced to occur during prototype testing.

#### B. DESCRIPTION OF SUBROUTINE ITCPLX

If the primary inverse iteration scheme fails to converge, control is passed to subroutine ITCPLX. The method of least squares is used to determine the two eigenvalues closest to the shift point, whether real or a conjugate pair. The eigenvector is not calculated. In the case of the conjugate pair the vector is of no interest (in this application) and in the case of two real eigenvalues a further refinement takes place which automatically includes determination of the two vectors.

#### C. LEAST SQUARES PROCEDURE

The method of least squares is a procedure for minimizing a function [Refs. 7, 9, 10]. Assume the solution to a problem is found by determining  $x$  and  $y$  in the following system.

$$\begin{aligned} f(x,y) &= a_1x + a_2y - a_3 = 0 \\ g(x,y) &= b_1x + b_2y - b_3 = 0 \end{aligned} \tag{C-1}$$

First form the function  $h(x,y) = f^2(x,y) + g^2(x,y)$ . Subsequently form the partial derivatives  $\left. \frac{\partial h}{\partial x} \right)_y$  and  $\left. \frac{\partial h}{\partial y} \right)_x$ . In the  $r^{\text{th}}$  step solve for  $x^r$  at the minimum with respect to  $x$  with the expression resulting from the first





partial derivative. Substitute this  $x^r$  in the expression resulting from the second partial derivative and solve for  $y^r$  at the minimum with respect to  $y$ . Substitute this  $y^r$  in the first expression and solve for  $x^{r+1}$ . Alternately solve for successive  $x^r$  and  $y^r$  until they meet some convergence criteria.

#### D. LEAST SQUARES IN MATRIX NOTATION

, System (C-1) may be written as

$$\begin{bmatrix} a_1 & a_2 \\ b_1 & b_2 \end{bmatrix} \begin{Bmatrix} x \\ y \end{Bmatrix} - \begin{Bmatrix} a_3 \\ b_3 \end{Bmatrix} = 0$$

and in general as

$$\begin{aligned} [a_{ij}] \{u_j\} - \{v_i\} &= 0 & i &= 1(1)m \\ & & j &= 1(1)n \end{aligned} \quad (C-2)$$

or as

$$A U - V = 0 \quad \text{where } m \text{ need not equal } n.$$

The least squares procedure may then be performed by forming

$$A^T A U - A^T V = 0 \quad \text{and solving for the } u_j.$$

By way of demonstration start with (C-2) and perform the matrix multiplication and rewrite as



$$\left\{ \sum_{j=1}^n a_{ij} u_j - v_i \right\} = 0 \quad i = 1(1)m$$

Form the function

$$h = \sum_{i=1}^m \left( \sum_{j=1}^n a_{ij} u_j - v_i \right)^2$$

and the partial derivatives

$$\left. \frac{\partial h}{\partial u_k} \right)_{\substack{u_j \\ j \neq k}} = 2 \sum_{i=1}^m a_{ik} \left( \sum_{j=1}^n a_{ij} u_j - v_i \right) = 0$$

for  $k = 1(1)n$ .

Delete the 2 as immaterial to the minimizing procedure

and write the  $\left. \frac{\partial h}{\partial u_k} \right)_{\substack{u_j \\ j \neq k}}$  as a system. Note that the row vector

$$[a_{ik}] = \{a_{ij}\}^T \quad \text{for } j = k.$$

$$\sum_{i=1}^m a_{ik} \left( \sum_{j=1}^n a_{ij} u_j - v_i \right)$$

$$= [a_{ik}] \left( \sum_{j=1}^n a_{ij} u_j - v_i \right) = 0$$

$$[a_{ik}] ([a_{ij}] \{u_j\} - \{v_i\}) = 0$$

$$[a_{ij}]^T [a_{ij}] \{u_j\} - [a_{ij}]^T \{v_i\} = 0$$

$$A^T A U - A^T V = 0$$



### E. SUBROUTINE PROCEDURE

Assume that the two eigenvalues closest to the shift point are the roots of the quadratic equation  $x^2 - ax - b = 0$ . ITCPLX determines  $a$  and  $b$  from which the roots may be extracted.

The equation is true for any vector  $Z = \{z\}$ .

$$(x^2 - ax - b) Z = x^2 Z - ax Z - b Z = 0$$

From the iteration process on the  $r^{\text{th}}$  iteration

$$xZ = \alpha^{r+1} Z^{r+1} \quad \text{where } \alpha^{r+1} \text{ is the maximum}$$

element in  $Z^{r+1}$  before scaling. It then follows that

$$\alpha^{r+2} \alpha^{r+1} Z^{r+2} - a \alpha^{r+1} Z^{r+1} - b Z^r = 0$$

Rewriting the above

$$a \alpha^{r+1} Z^{r+1} + b Z^r = \alpha^{r+2} \alpha^{r+1} Z^{r+2}$$

and

$$[Z] \begin{Bmatrix} \alpha^{r+1} & a \\ & b \end{Bmatrix} = \alpha^{r+2} \alpha^{r+1} Z^{r+2}$$

where

$$[Z] = [Z^{r+1} \ Z^r]$$



Performing the least squares procedure

$$[Z]^T [Z] \begin{Bmatrix} \alpha^{r+1} & a \\ & b \end{Bmatrix} = \alpha^{r+2} \alpha^{r+1} [Z]^T \{Z\}^{r+2}$$

Moving the  $\alpha^{r+1}$  to the first column and rewriting

$$\begin{bmatrix} \alpha^{r+1} & Z^{r+1T} & Z^{r+1} & Z^{r+1T} & Z^r \\ \alpha^{r+1} & Z^rT & Z^{r+1} & Z^rT & Z^r \end{bmatrix} \begin{Bmatrix} a \\ b \end{Bmatrix} \\ = \alpha^{r+1} \alpha^{r+2} \begin{Bmatrix} Z^{r+1T} & Z^{r+2} \\ Z^rT & Z^{r+2} \end{Bmatrix}$$

The  $\alpha^{r+1}$  and  $Z^{r+1}$  are solved for by Potters' Gaussian elimination procedure. The two eigenvalues are then  $s + t$  and  $s - t$  where  $s = \frac{a}{2}$  and  $t = \frac{\sqrt{a^2 + 4b}}{2}$ . The test for two real roots versus a complex conjugate pair is the sign of the discriminant  $a^2 + 4b$ . The two convergence tests are

$$\left| \frac{|s^{r+1}| - |s^r|}{s^{r+1}} \right| \quad \text{less than EPSVIB}$$

and

$$\left| \frac{|t^{r+1}| - |t^r|}{t^{r+1}} \right| \quad \text{less than EPSVIB.}$$





## F. SUBROUTINE FORTRAN VARIABLES

In subroutine IPCPLX the Fortran Variables are

$$\begin{array}{ll}
 \text{SCALEW} &= \alpha^{r+2} & \text{WT(I)} &= z^{r+2} \\
 \text{SCALEX} &= \alpha^{r+1} & \text{XT(I)} &= z^{r+1} \\
 \text{SCALE3} &= \alpha^r & \text{YT(I)} &= z^r \\
 \text{ZA} &= a & \text{ZB} &= b \\
 \text{VREAL} &= s^{r+1} & \text{VIMAG} &= t^{r+1} \\
 \text{OLDVR} &= s^r & \text{OLDVI} &= t^r
 \end{array}$$

The system set-up is

$$\begin{bmatrix} \text{B} & \text{C} \\ \text{E} & \text{F} \end{bmatrix} \begin{Bmatrix} \text{ZA} \\ \text{ZB} \end{Bmatrix} = \begin{Bmatrix} \text{D} \\ \text{G} \end{Bmatrix}$$

where

$$\text{B} = \text{SCALEX} * \left( \sum_{\text{I=IGO}}^{\text{ISTOP}} \text{XT(I)} * \text{XT(I)} \right)$$

and

$$\text{G} = \text{SCALEX} * \text{SCALEW} * \left( \sum_{\text{I=IGO}}^{\text{ISTOP}} \text{YT(I)} * \text{WT(I)} \right)$$

etc.....

and

$$\text{VREAL} = 0.5 * \text{ZA}$$

$$\text{VIMAG} = 0.5 * \text{DSQRT} (\text{ZA} * \text{ZA} + 4.0 * \text{ZB})$$



## G. PERFORMANCE OF SUBROUTINE ITCPLX

As mentioned before, ITCPLX was never invoked by SATANS-III during the execution of the test cases. However, during prototype testing it was possible to evaluate the procedure extensively [Refs. 7, 8, 16 and 17]. In all cases, whenever the inverse iteration scheme failed subroutine ITCPLX returned the correct eigenvalues, both real and complex. This subroutine provides an excellent backup and diagnostic tool for the basic solution procedure.

Since the subroutine uses Potters' scheme to determine the  $z^{r+1}$  it is subject to the same general ill-conditioning problem that the primary scheme is subject to in this respect. A second problem arises when  $t^r$  is extremely small. This is a highly unlikely case in practice in regards to this application. In the case of total non-convergence by both schemes the results of the last two iterations of both schemes are printed along with the information on whether they would be real or complex values. This information is valuable in analyzing a difficult problem. The flowchart for subroutine ITCPLX is depicted in Figure 13.

## V. SUBROUTINE RANGER

### A. FREQUENCY RANGE SEARCH

The primary objective in a frequency range search operation is to provide some method of ensuring that all the frequencies in the given range are determined or, if missed, determining which are missed. This is accomplished



in two ways in SATANS-III. Each eigenvalue is determined three times in covering a range. For each eigenvalue determined, the mode shape is simultaneously determined and printed once, along with the eigenvalue (frequency). Determining an eigenvalue three times is time consuming but reduces the chance of missing a mode and examining the mode shapes determined is an easy way of spotting a missed mode.

The method employed uses the basic iteration scheme to actually determine the eigenvalues. No a priori knowledge of the number or distribution of frequencies in the range is assumed. The amount of work involved is directly proportional to the number, separation and distribution of frequencies in the range and the operating parameters.

The calculations involve two primary subroutines, RANGER and ADJUST. Subroutine RANGER contains the primary search scheme and initial eigenvalue determination scheme. Two other subroutines are involved, BTM and TOPPER, which determine the lowest and highest frequencies in the range, respectively.

The only additional inputs by the user for this operation are the frequency range endpoints in Hz.

#### B. SEARCH SCHEME

The eigenvalues naturally determined by the system of four second order equations are negative ( $-\omega^2 T_o^2$ ). It is easier to search the field of negative numbers rather than alter the set-up of the problem to yield positive numbers. For this reason the lower bound of the range,



BOTTOM, is the negative number smallest in magnitude and corresponds to the lowest frequency. Similarly the upper bound of the range, TOP, is the negative number largest in magnitude and corresponds to the highest frequency.

For example if the range from 1.59 Hz to 5.29 Hz is to be searched then  $BOTTOM = -(6.28(1.59)1.0)^2 = -100$  where  $6.28 = 2 \text{ PI}$  and  $1.0 = T_0$  and  $TOP = -(6.28(5.29)1.0)^2 = -1,100$ . These two numbers establish the initial range. This initial range is subsequently partitioned into a number of smaller ranges as new eigenvalues are determined, and each smaller range is eliminated when all eigenvalues within that range have been determined.

Each range has associated with it four parameters stored in the array RANGE (100,4). For the sake of clarity the elements of this array will be referred to as  $R_1$  in the example that follows. For a given range, K, the elements of the array RANGE are

#### EXAMPLE

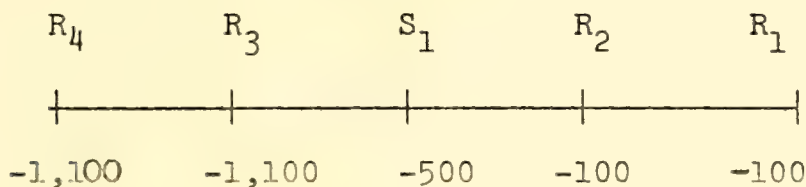
1. Lowest eigenvalue in the range,  $R_1 = -100$ .  
RANGE(K,1)
2. Either the lowest eigenvalue or  $R_2 = -100$ .  
the maximum value in the band-  
width from which it was  
determined. RANGE(K,2)





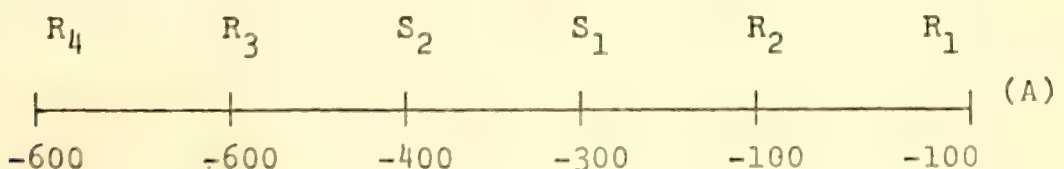
3. Either the highest eigenvalue  $R_3 = -1,100.$   
or the minimum value in the  
bandwidth from which it was  
determined.  $RANGE(K,3)$
4. Highest eigenvalue in the  $R_4 = -1,100.$   
range.  $RANGE(K,4)$

Therefore,  $RANGE(K,J)$  for  $J = 1(1)4$  is  $(-100., -100., -1,100., -1,100.)$ , and is represented graphically as

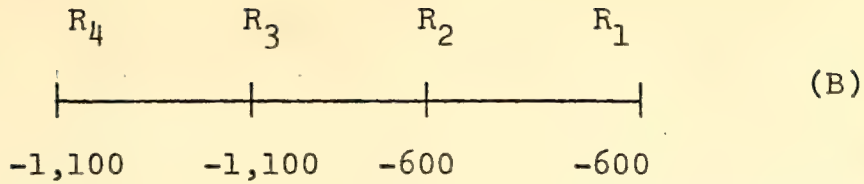


The range of frequencies is searched by choosing a suitable spectral shift point. (Present program parameters call for 40% up from the bottom of the range specified by  $R_2$ . This may be a user specified parameter by altering the statement  $TOLCLS = .4$  in subroutine INITAA). In the example the first shift point is  $-500$ ,  $S_1$ .

Assume a new eigenvalue,  $-600$ , is determined. The initial range is then partitioned in two ranges as shown below.

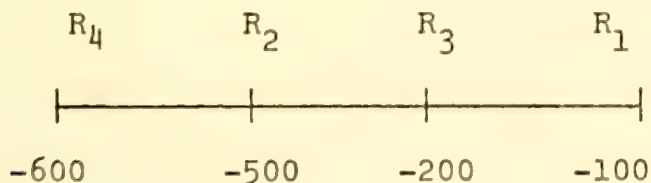






The above partitioning occurs each time a new eigenvalue is determined. The next shift point,  $S_1$ , is -300.

Assume that there are no more eigenvalues in the range from -100 to -600. It is the nature of the inverse iteration process that the eigenvalue closest to the spectral shift point will be determined. This is -100 in the example. Because of this property it is assumed that no other eigenvalue exists within  $\pm 200$  of  $S_1$ . (Otherwise it should have been determined.) If the eigenvalue -100 is determined from the shift point -300 then  $R_2$  is set to the maximum value of the bandwidth.  $R_2 = -300 + (-300 - (-100)) = -500$ . The next shift point is  $S_2 = -400$  (40% down from the top.). From the spectral shift point the higher eigenvalue, -600, will be determined and a similar calculation for  $R_3$  is performed;  $R_3 = -400 - (-600 - (-400)) = -200$ . The following situation then exists



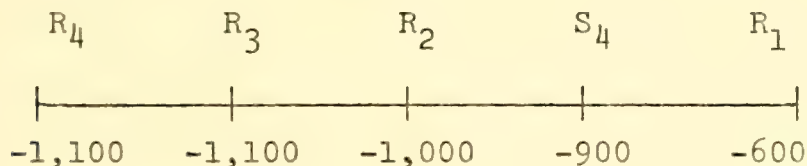


This is how a range is eliminated as being completely covered. The range elimination test is

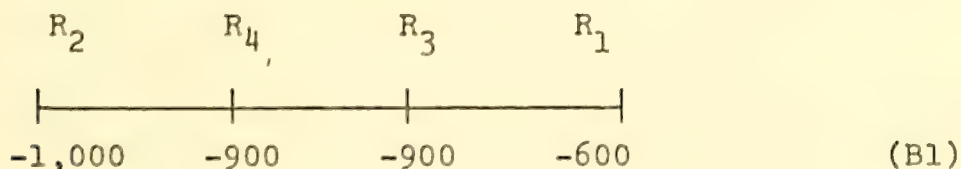
$$\text{Is } R_2 = < \text{TOLRNG} * R_3 ? \quad (\text{TOLRNG} = 1.0 - (.4 * \text{EPS}))$$

and is intended to prevent missing a mode near the center of the range when a shift point of 25% from the top or bottom is used. (TOLRNG is set in subroutine INITAA.)

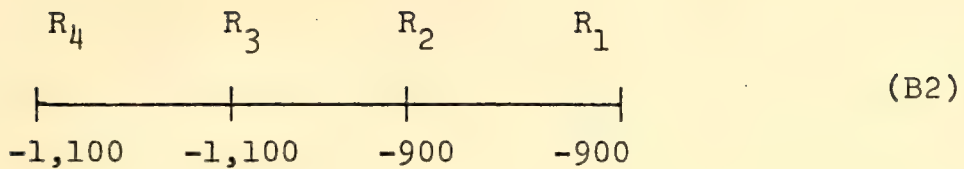
The first shift point,  $S_3$ , in the second range, (B), is -800, and assume further that the lower eigenvalue of -600 has been determined (for the third time !!!). The situation is as follows



The next shift point (40% down from the top as specified by  $R_3$ ) is  $-900 = S_4$ . From this spectral shift point assume that an eigenvalue of -900 is determined. The partition would then be







with range (B1) being immediately eliminated. The process would be repeated with range (B2).

A range, or portions thereof, may be eliminated because no real eigenvalue is found within the range, or because an eigenvalue outside the range is found. Consideration must also be given to the situation where ill-conditioning would cause the primary scheme to fail and two real eigenvalues might be determined by subroutine ITCPLX.

In deciding if an eigenvalue has been previously determined when eliminating a range, consideration should be given to the desired results of the problem. For example, is it necessary to know all frequencies in a range even if four of them differ by less than 1%? A set of parameters (TOLLOW and TOLHI) determine the degree of separation that defines individual eigenvalues and are presently set at  $\pm 1/2\%$ . They are set in subroutine INITAA. Thus, if two eigenvalues differ by less than  $\pm 1/2\%$  they are defined as one eigenvalue.

Subroutine RANGER has incorporated an Aitkin extrapolation to accelerate the convergence rate of the eigenvalues. To further reduce computation time the convergence criteria, EPSAIT, on the accelerated





eigenvalues is different from the criteria, EPSVIB, for the refined eigenvalues. EPSAIT should be commensurate with the chosen separation criteria. EPSAIT is currently set at .01 in subroutine INITAA.

Since the lowest frequencies are of most interest the search scheme is biased toward the lower frequencies. The shift points chosen first are in the lower portion of the range. Second is the choice of vectors. For the first vector the initial guess is all ones. Subsequently the guess is the difference between the  $r^{\text{th}}$  and  $r-2$  iterated vector. The iterated vectors are richer in the lower modes than the higher modes. (Consider the expansion of an arbitrary vector in the set of orthogonal eigenvectors.) This amounts to a pseudo sweeping technique. The actual calculation of the difference of the two vectors is performed in subroutine ITRATE. Figure 9 depicts the search scheme schematic. The flowchart for subroutine RANGER is depicted in Figure 14.

## VI. SUBROUTINE ADJUST

Subroutine ADJUST is the second of the subroutines involved in the frequency range search operation. (The first was subroutine RANGER.) Three main functions are performed by this subroutine.

A. Newly determined eigenvalues and eigenvectors are refined.

B. Range partition and elimination calculations are



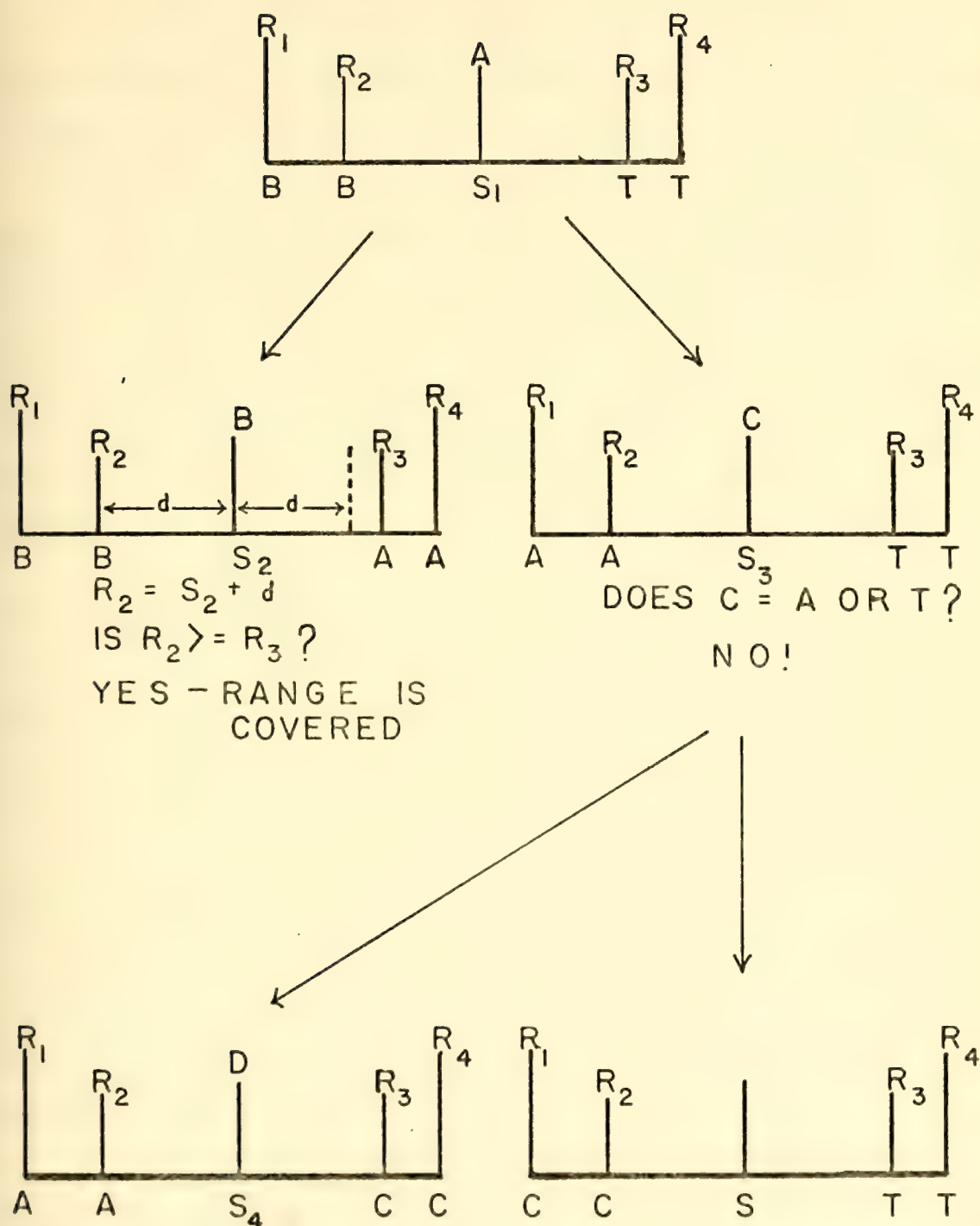


FIGURE 9 SEARCH SCHEME SCHEMATIC



performed including the range elimination test.

C. The situation of a complex conjugate pair or two real eigenvalues, both in the range, only one in the range or both outside the range, is handled in this subroutine.

In refining a newly determined eigenvalue, a new shift point close to the subject eigenvalue is chosen. How close is determined by the parameter TOLSHF (currently set at 90%) set in subroutine INITAA. A new vector is not obtained and the iteration begins with the last obtained vector. Figure 15 depicts the flowchart for subroutine ADJUST.

## VII. SUBROUTINE BTM

Subroutine BTM is one of two minor subroutines (BTM and TOPPER) called in subroutine RANGER. The function of BTM is to determine the minimum eigenvalue of the range. The initial shift point used is the lower bound specified by the user in his input statement.

If a real eigenvalue cannot be found at the first shift point a new shift point 10% up the range is tried. This continues until the range is completely covered or a real eigenvalue is determined.

A form of the Aitkin extrapolation is used to accelerate the initial convergence rate.

The vector convergence test is always performed in this subroutine in determining the minimum eigenvalue. The flowchart for subroutine BTM is depicted in Figure 16.



#### VIII. SUBROUTINE TOPPER

Subroutine TOPPER operates in a manner analogous to subroutine BTM in determining the maximum eigenvalue of the range.

The main differences are the initial shift point, which is the upper bound specified by the user, and the fact that the vector convergence test is not performed unless specified by the user. The flowchart for subroutine TOPPER is depicted in Figure 17.

#### IX. SUBROUTINE INITAA

Subroutine INITAA, called only in a vibration analysis, sets the operating parameters of the program.

IBEGIN, IEND, IGO and ISTOP are do-loop begin and terminate parameters to ease handling of the eigenvectors as doubly and singly dimensional arrays.

The rest of the program is self-explanatory. See the program listing. Figure 18 depicts the flowchart for subroutine INITAA.

#### X. SUBROUTINE START

Subroutine START initializes iteration parameters and the starting vector. The flowchart for subroutine START is depicted in Figure 19.

#### XI. SUBROUTINE VECOUT

This subroutine controls the printed output for the vibration analyses. Several operations are performed in





addition to printing output.

A. The eigenvalues are converted to real time cycles per second (Hz).

B. A check is made for imaginary frequencies.

C. The double precision solution vector WV(I) is mapped into single precision space prior to calling subroutine OUTPUT which prints the solution vector.

Examples of all possible printed messages are presented in Figure 20. Lines in braces are optional or alternate possibilities.



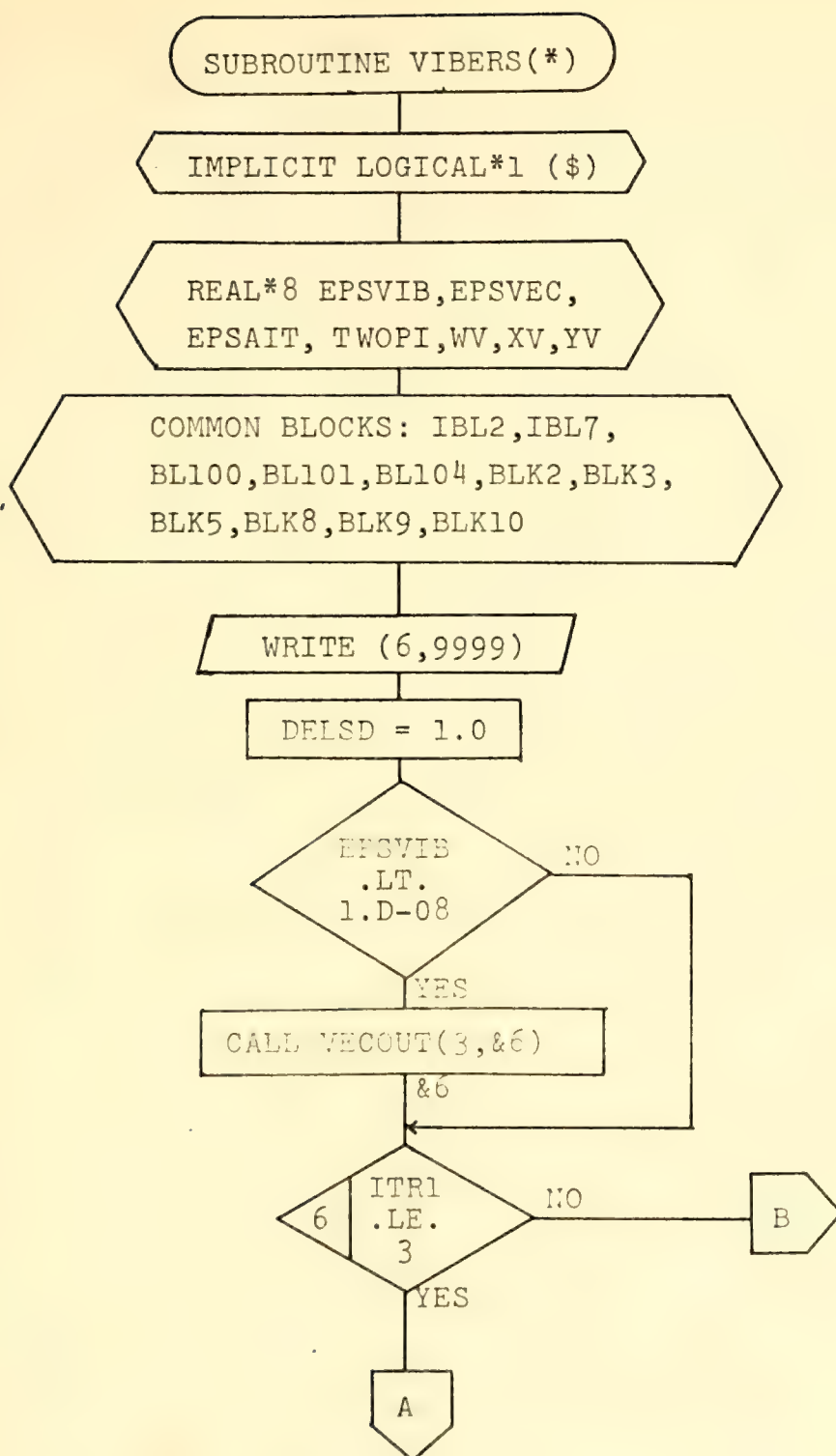


FIGURE 10. SUBROUTINE VIBERS(\*) FLOWCHART



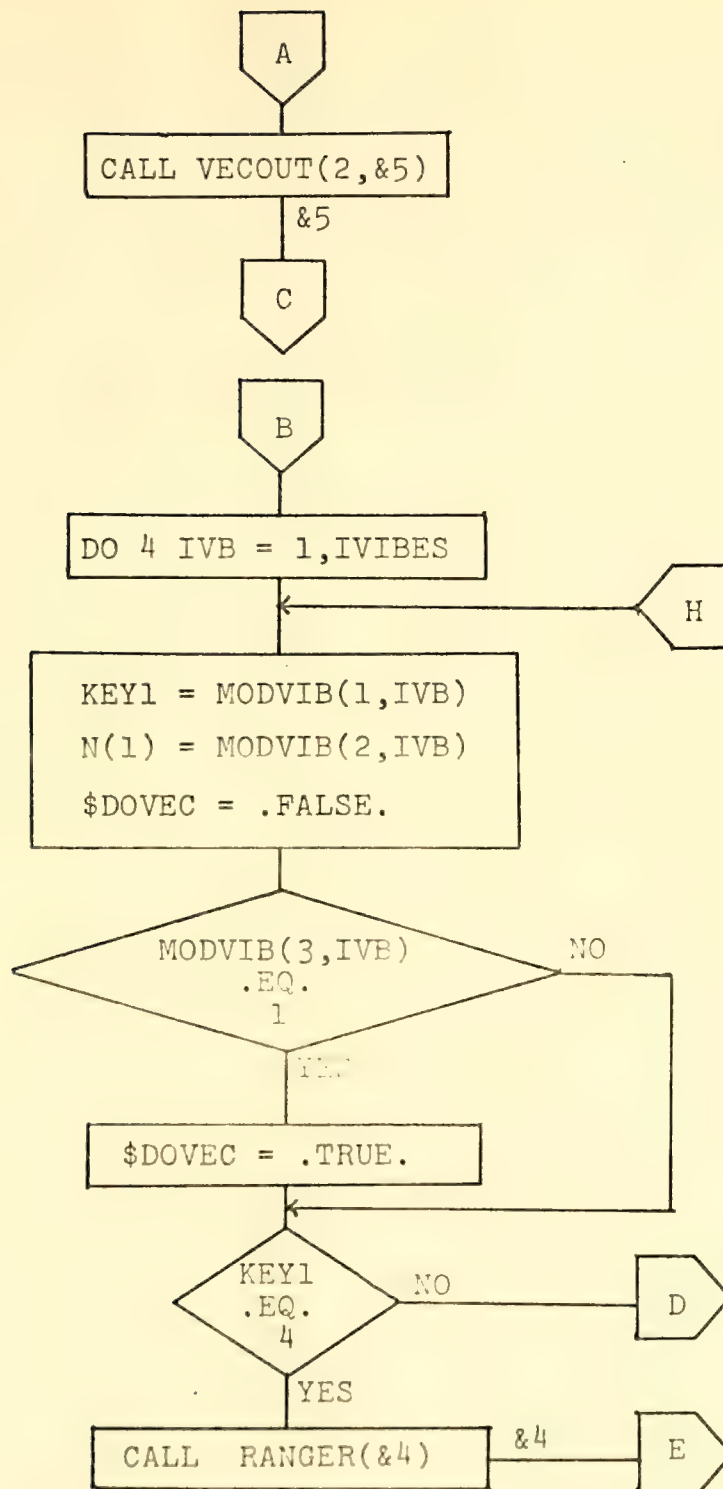


FIGURE 10 continued



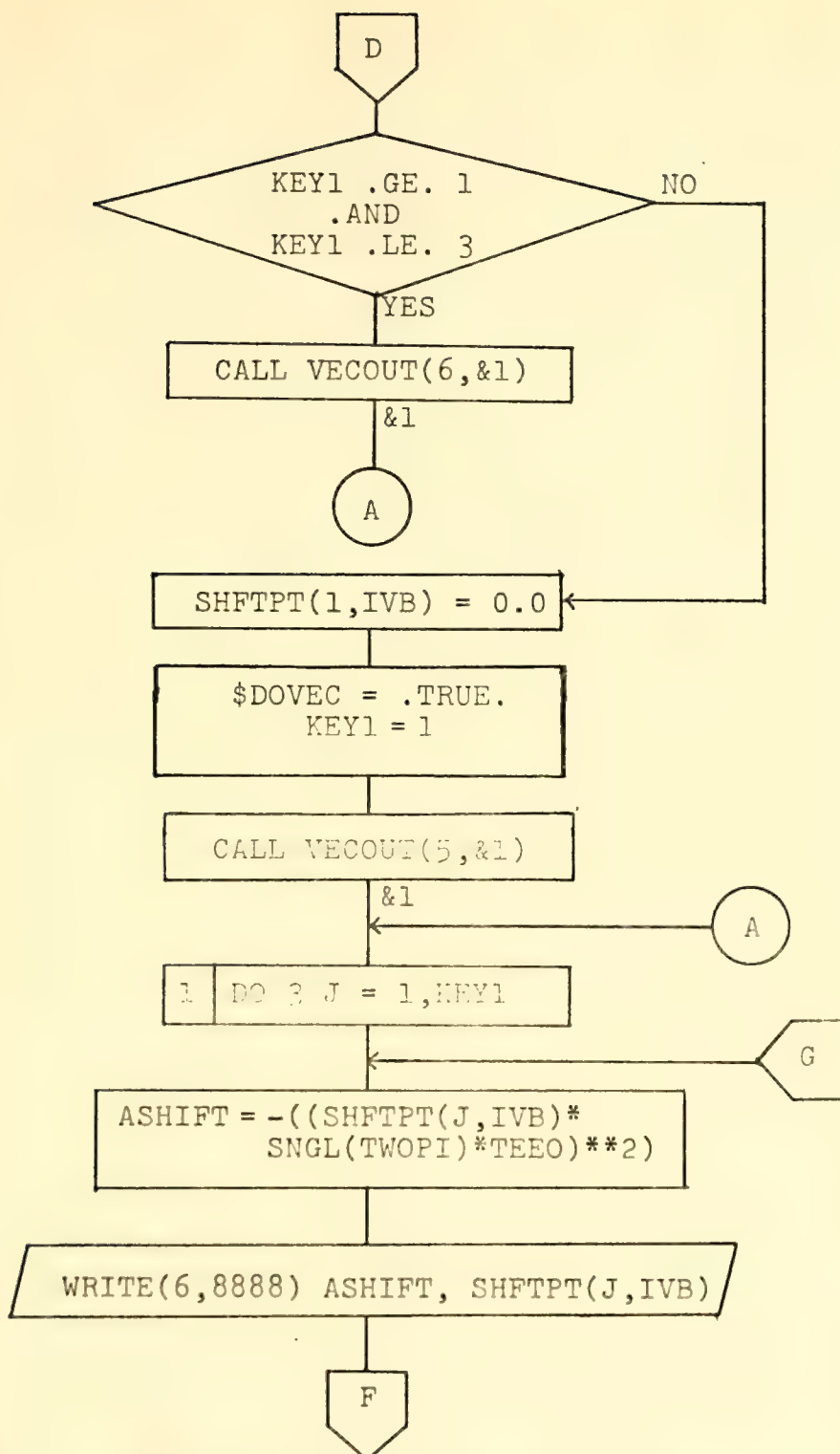


FIGURE 10 continued





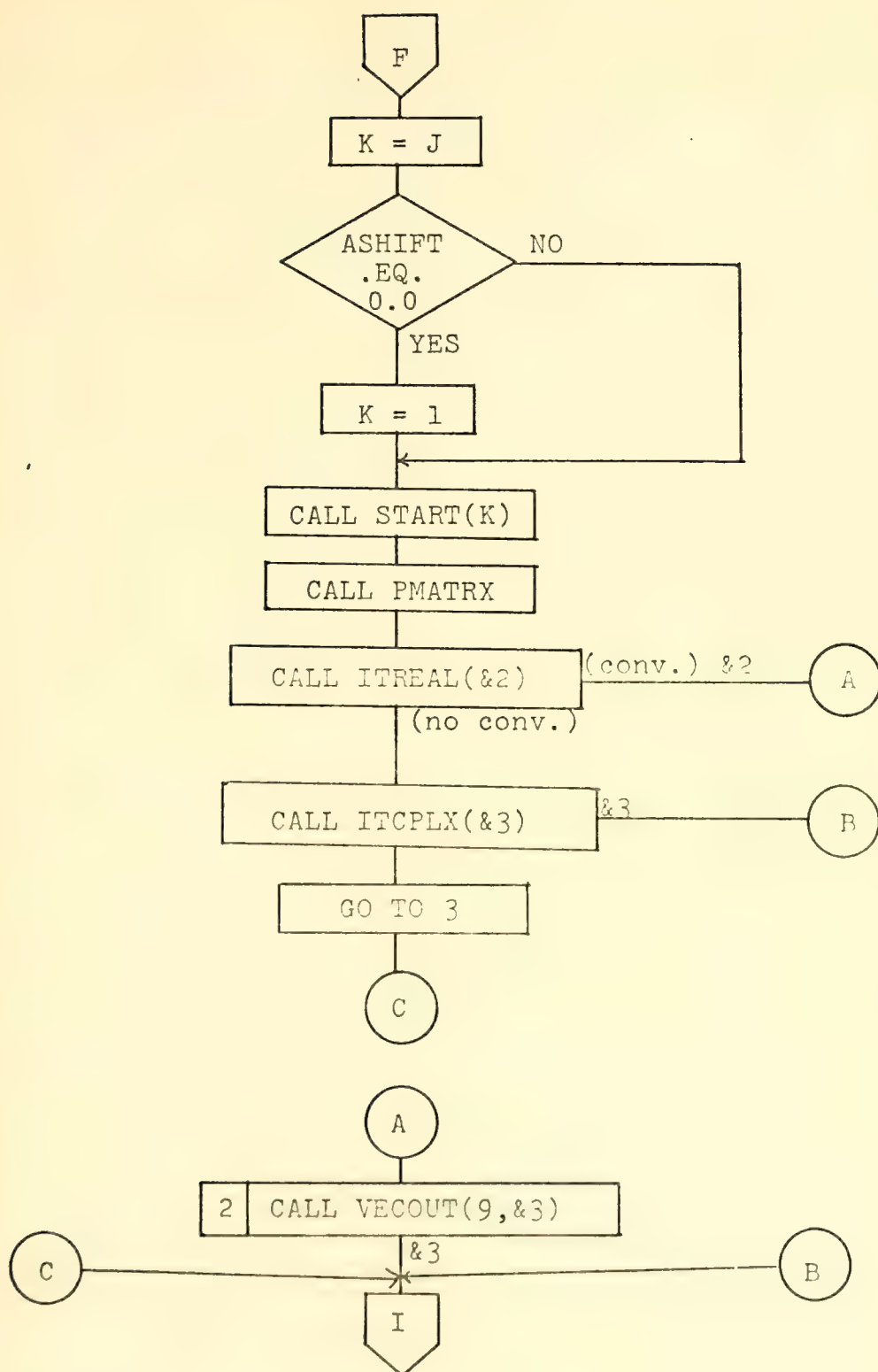


FIGURE 10 continued



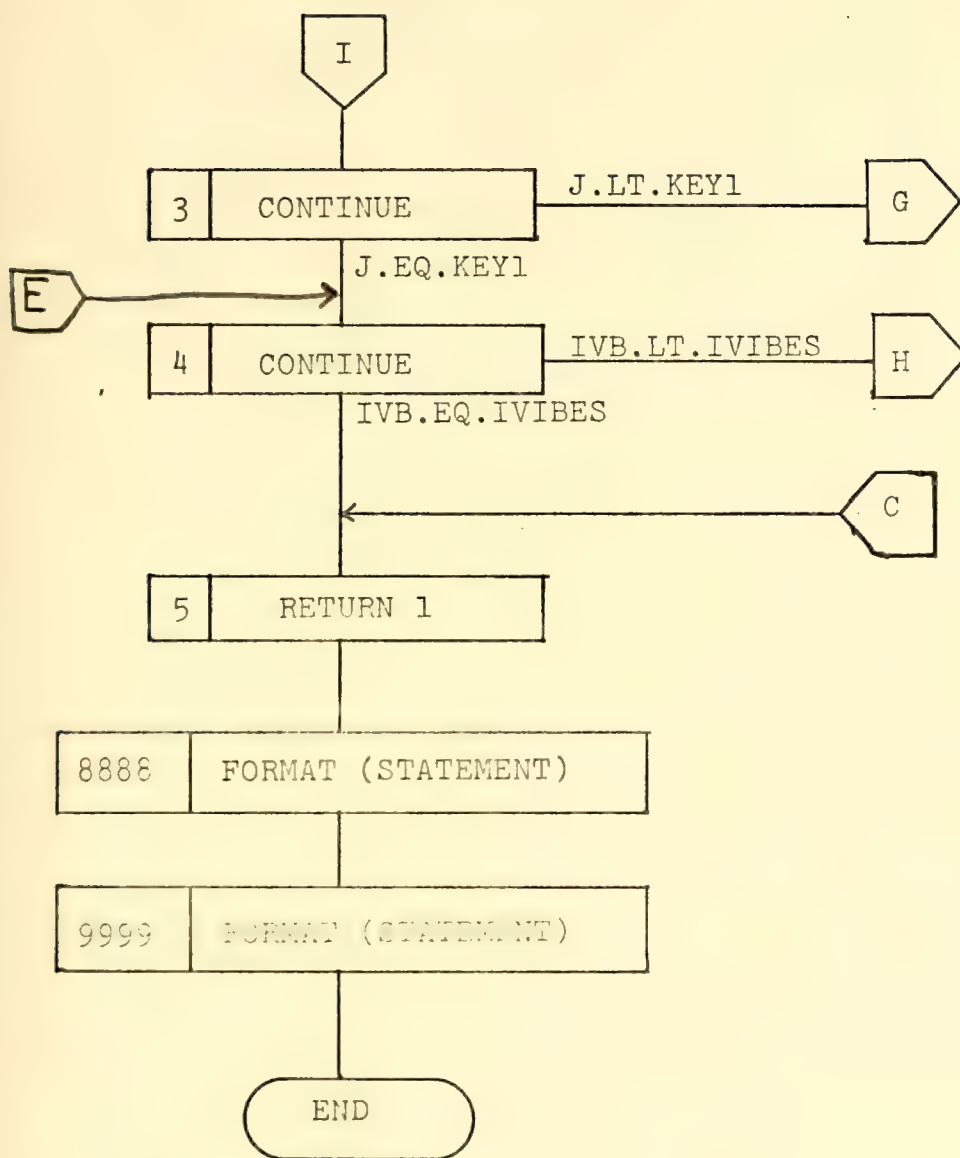


FIGURE 10 continued



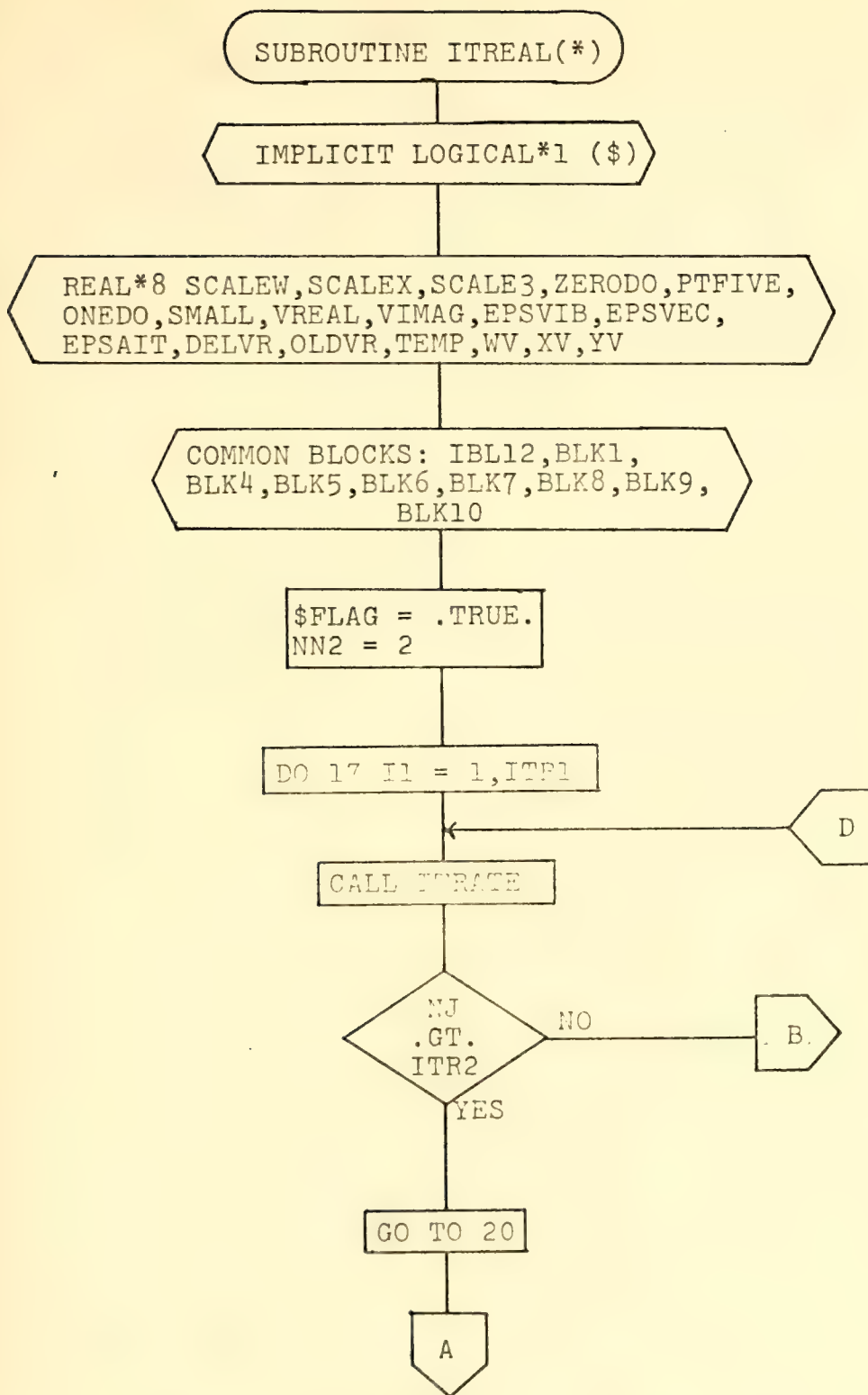


FIGURE 11. SUBROUTINE ITREAL(\*) FLOWCHART



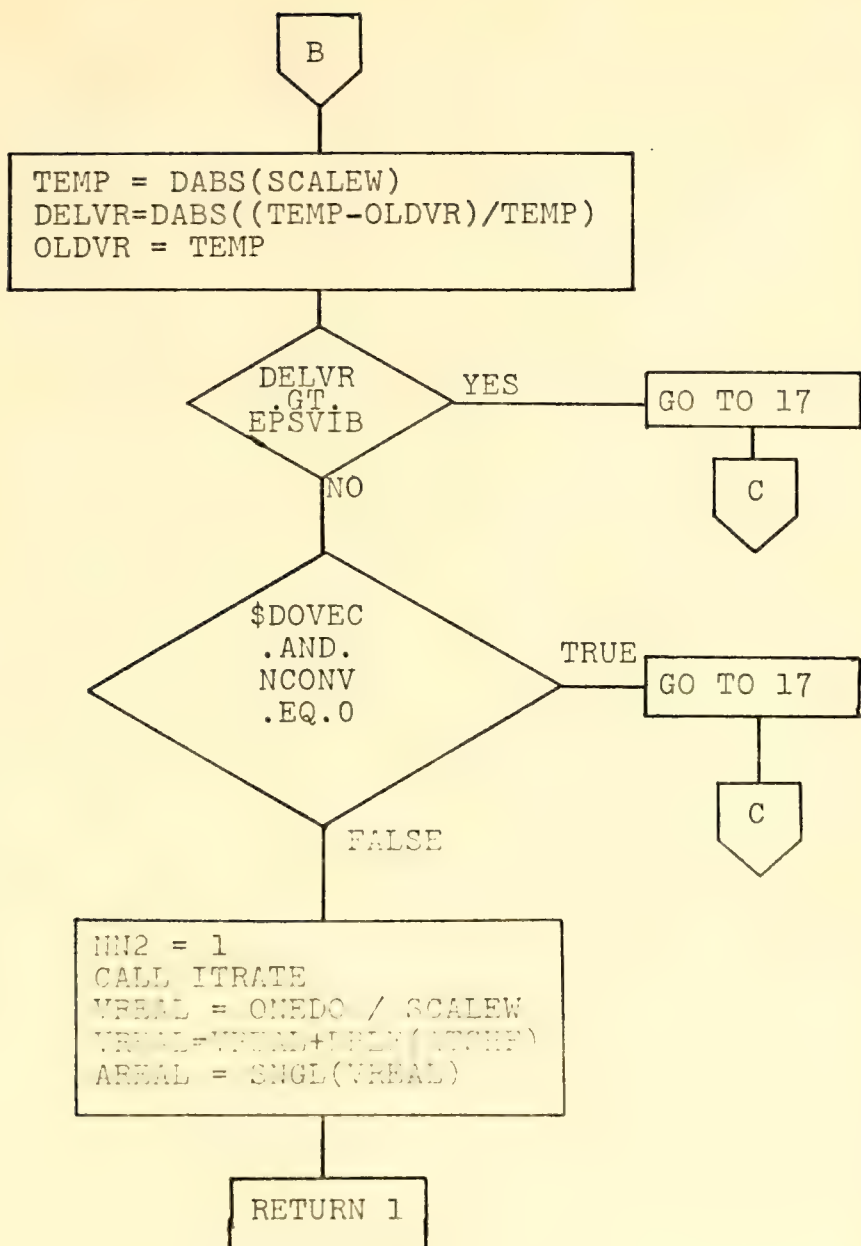


FIGURE 11 continued





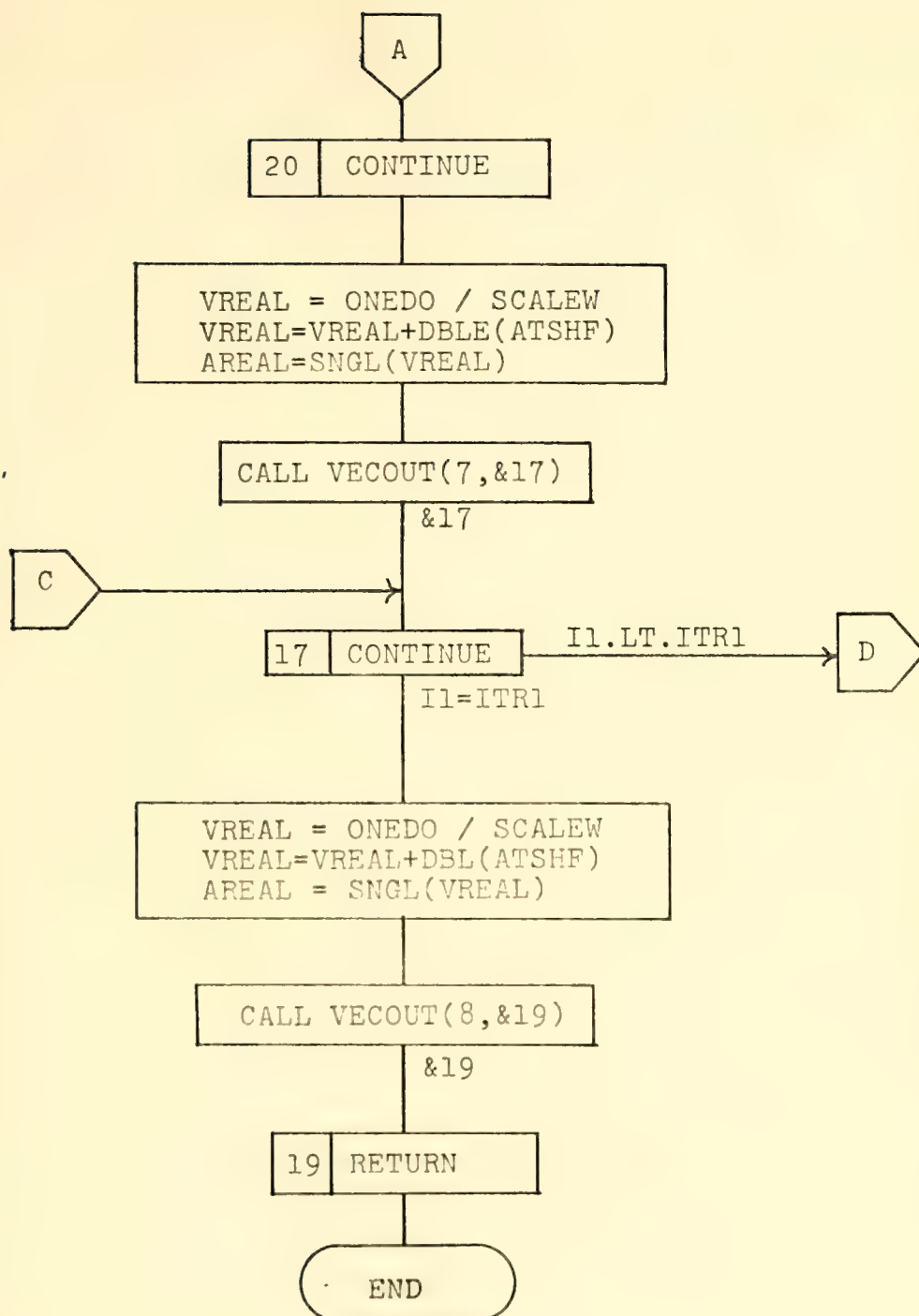


FIGURE 11 continued



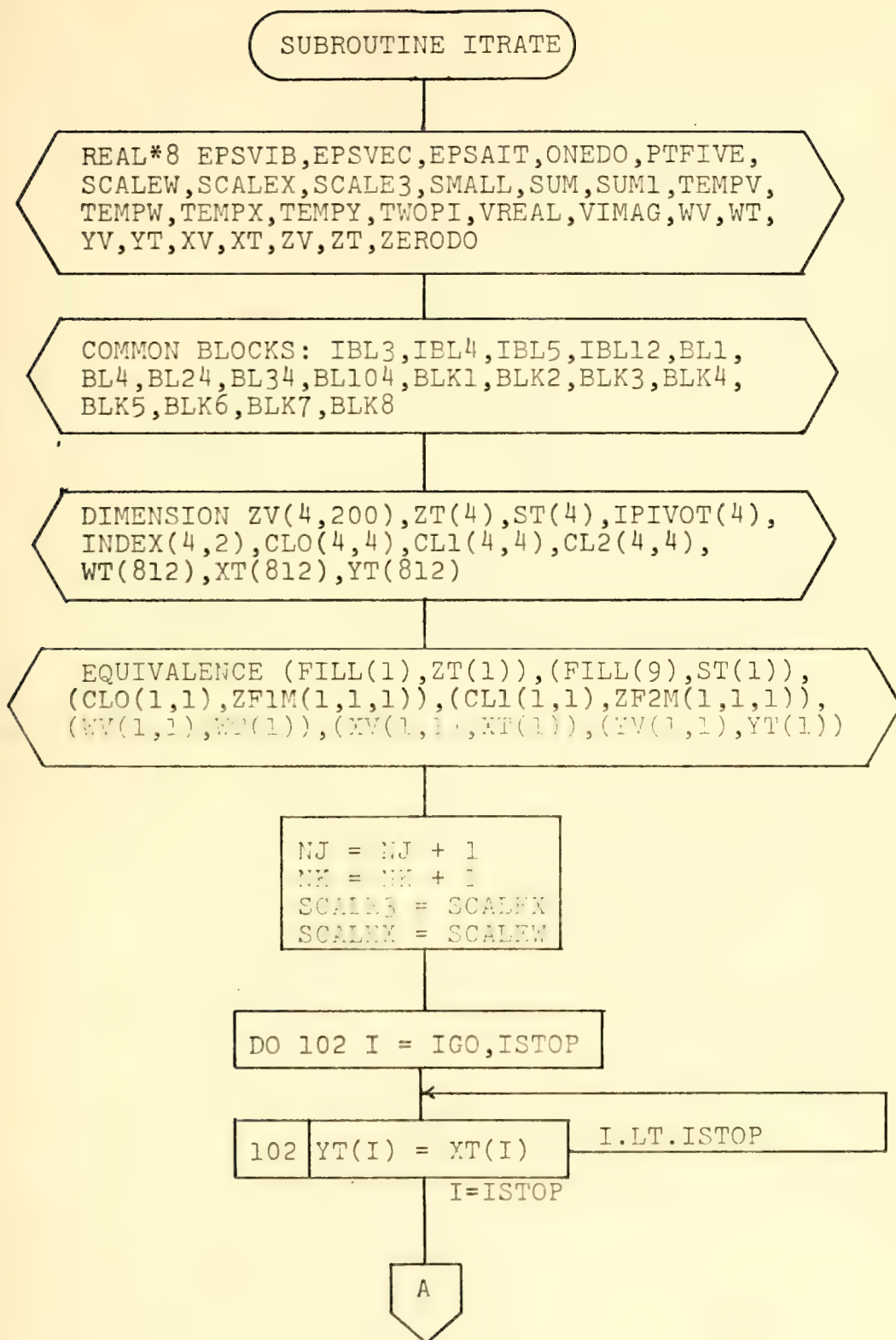


FIGURE 12. SUBROUTINE ITRATE FLOWCHART



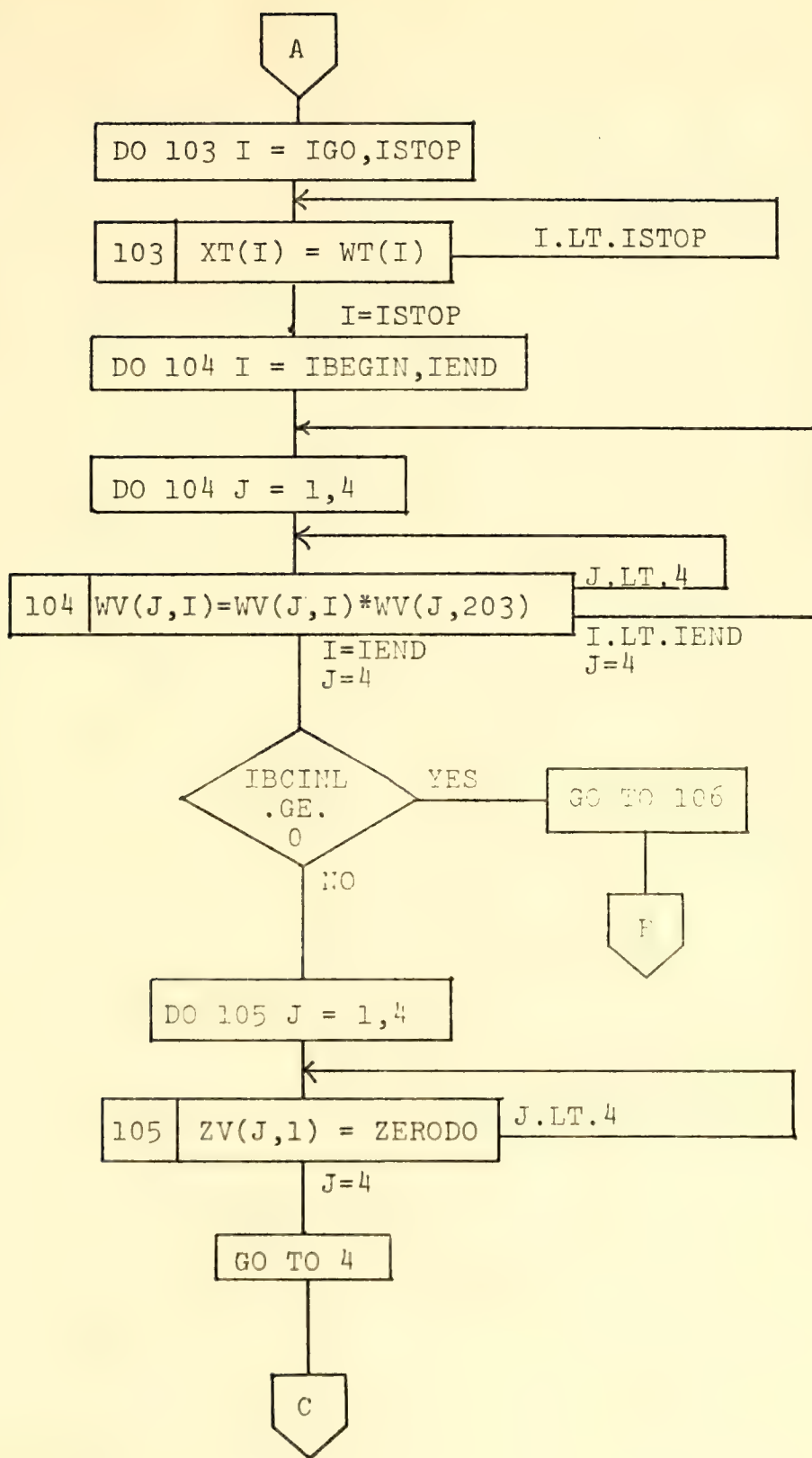


FIGURE 12 continued



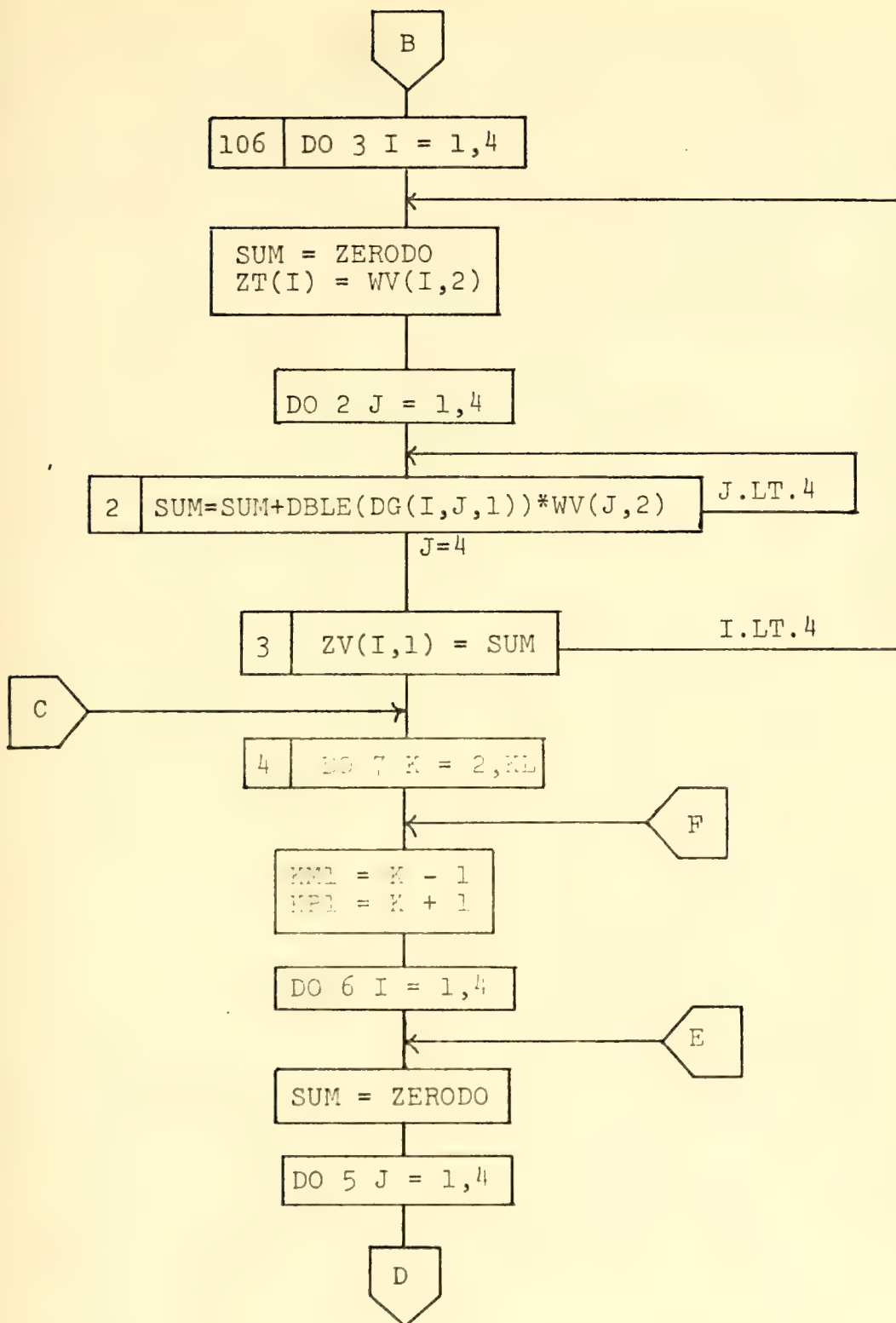


FIGURE 12 continued





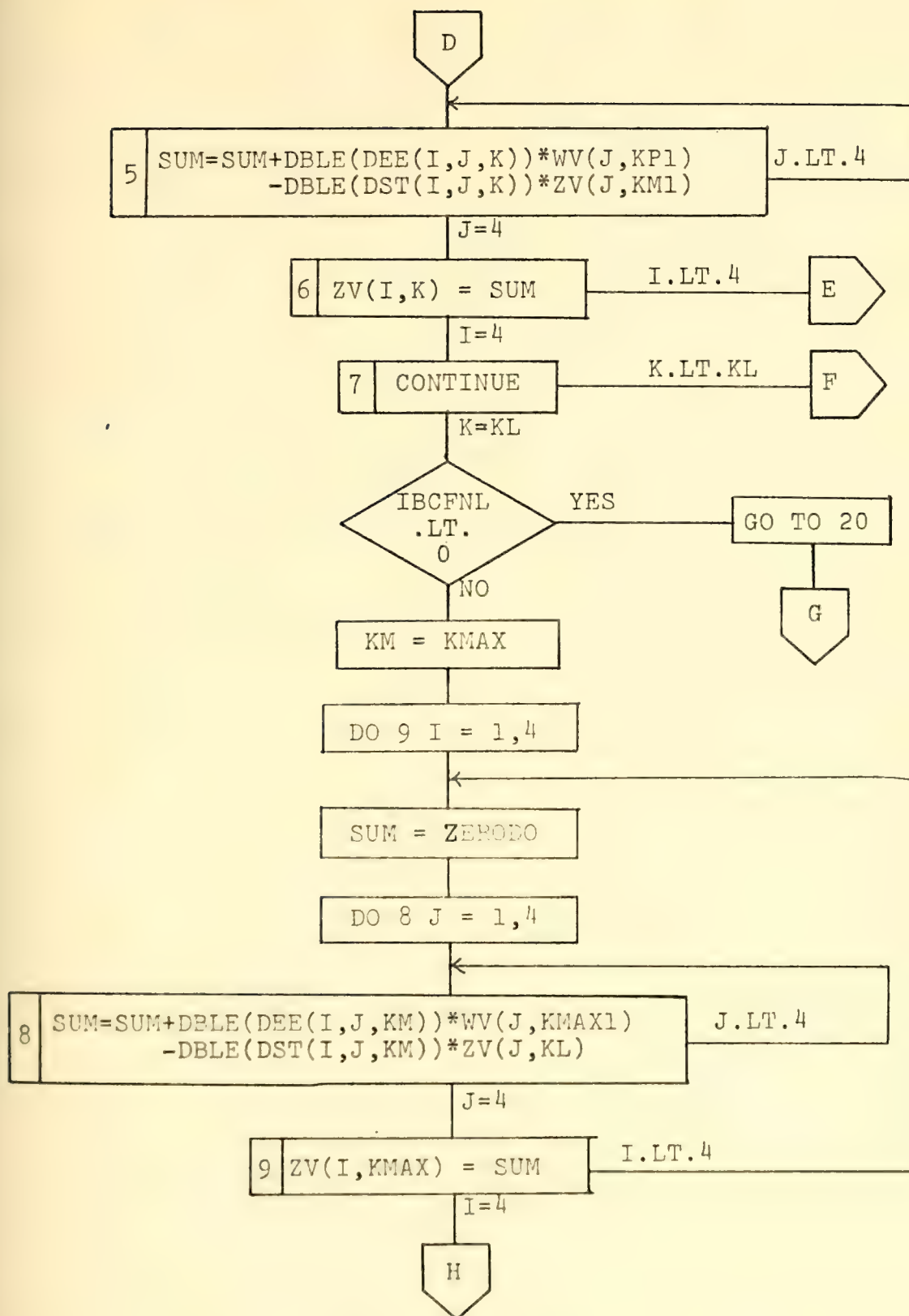


FIGURE 12 continued



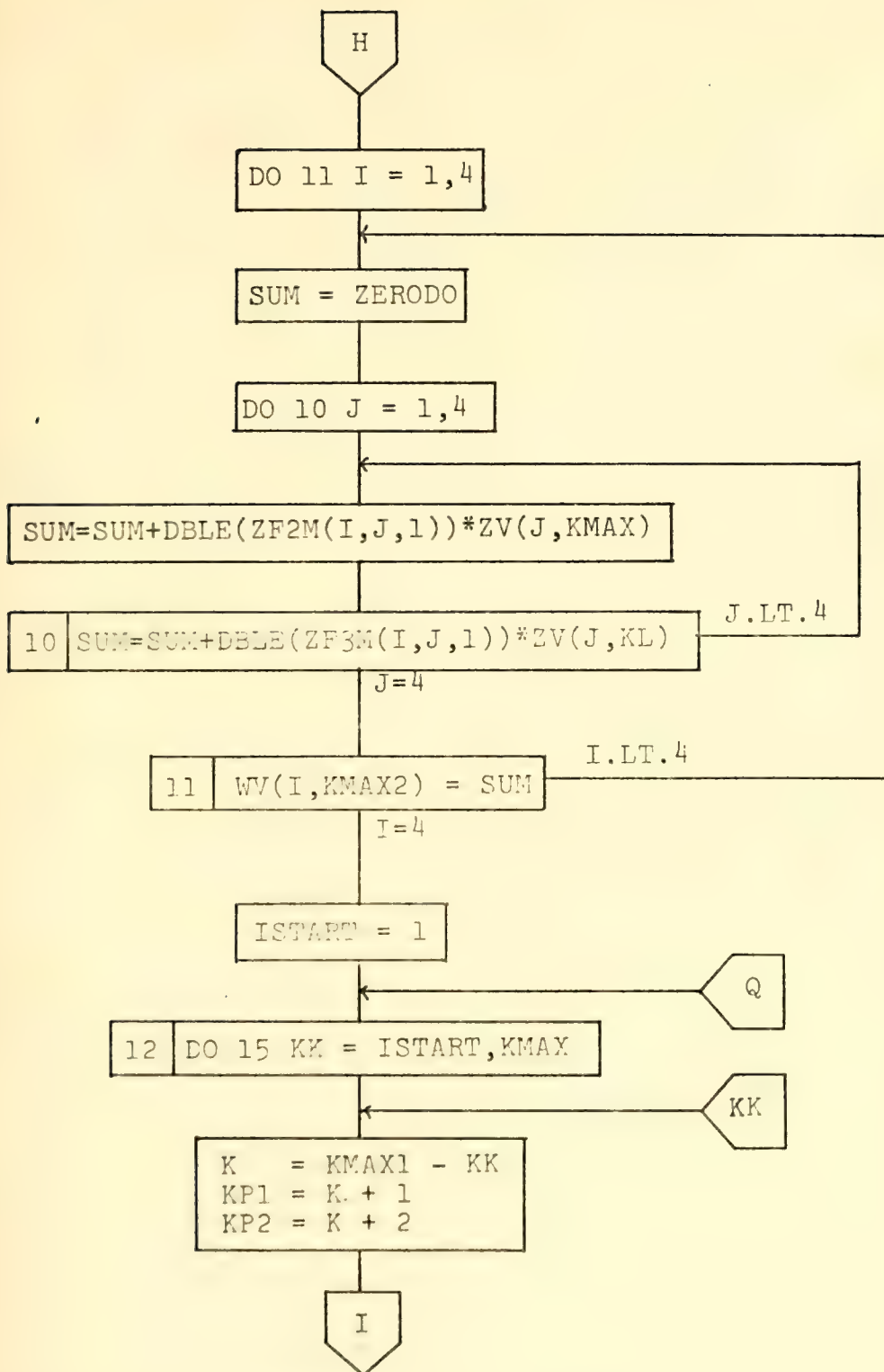


FIGURE 12 continued



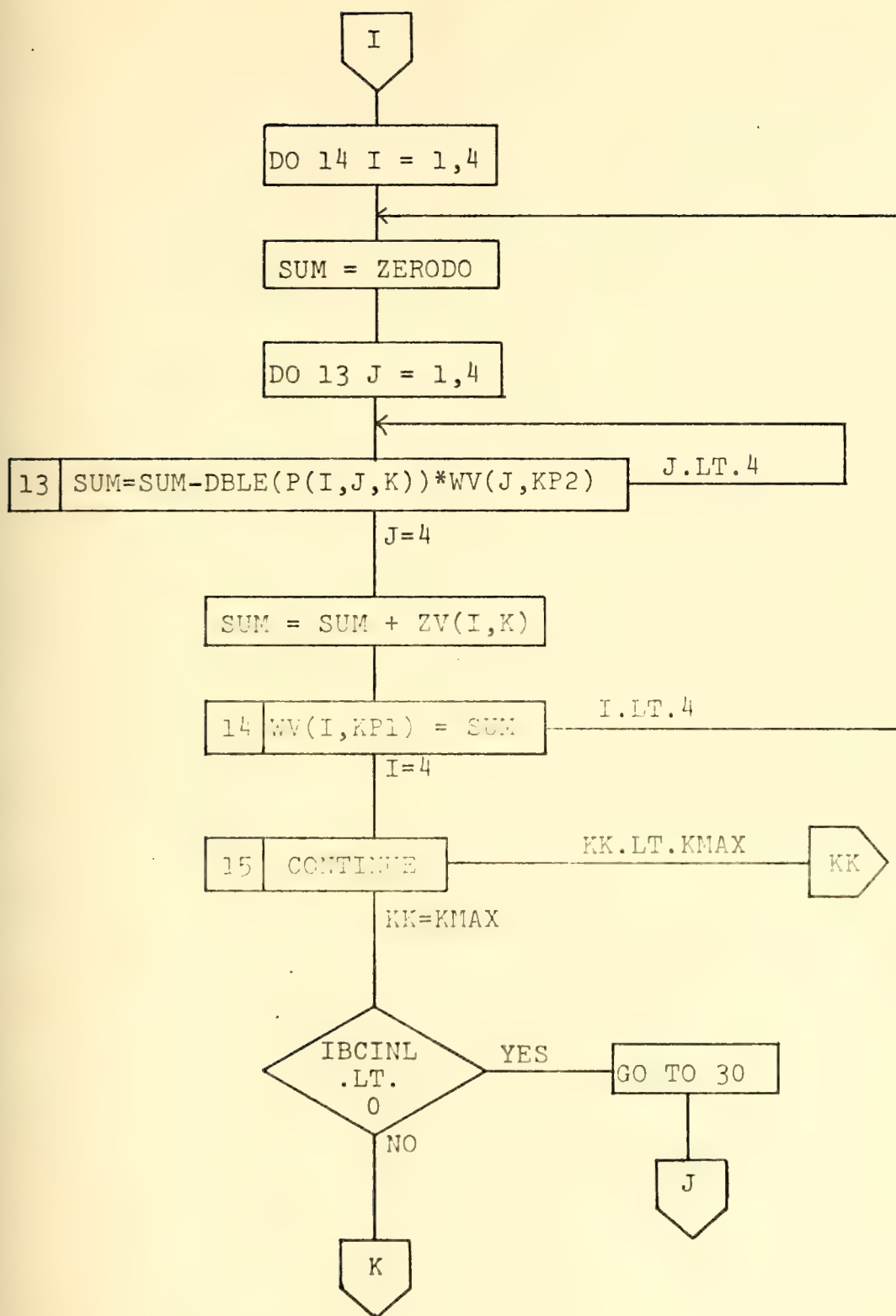


FIGURE 12 continued



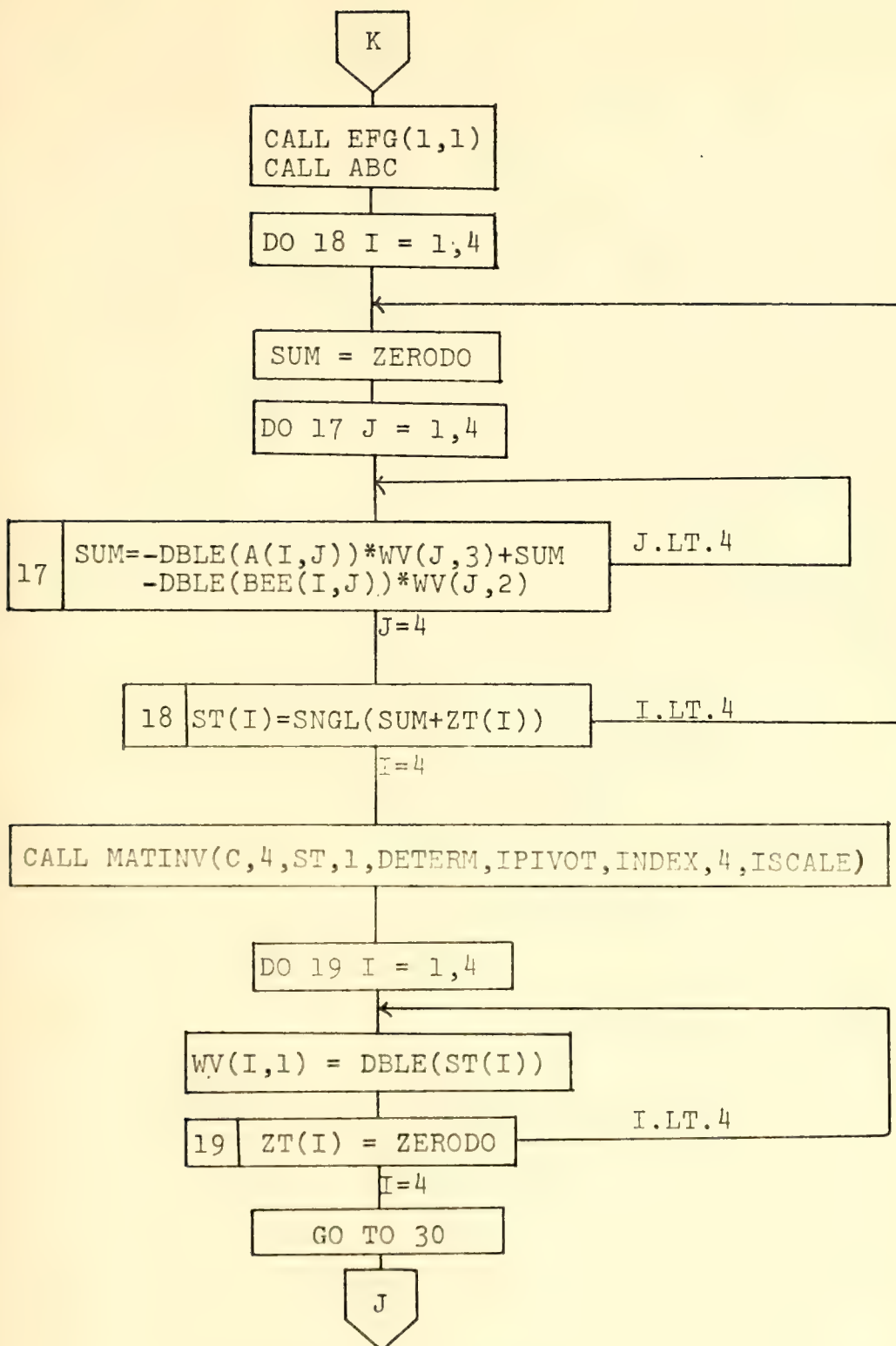


FIGURE 12 continued





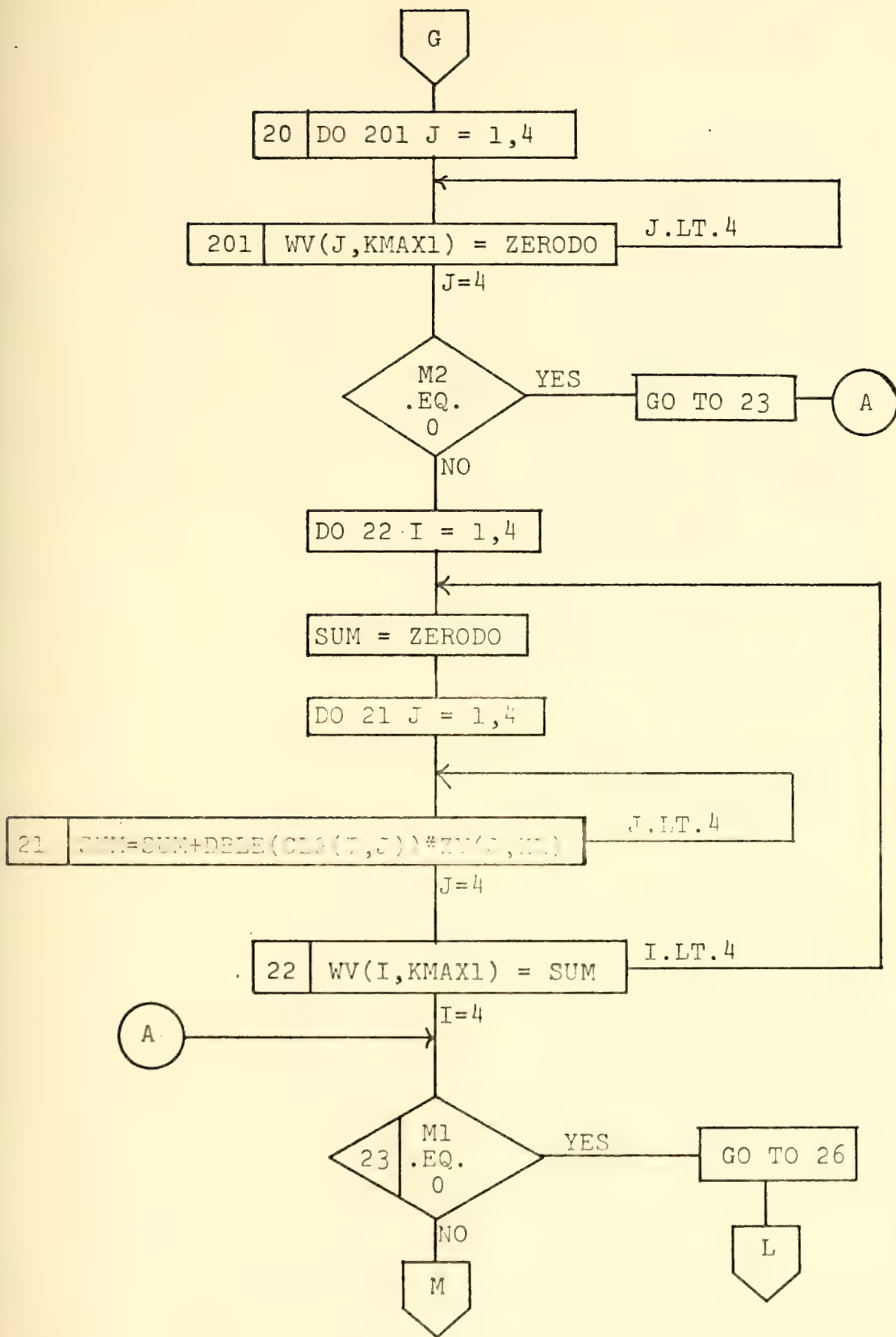


FIGURE 12 continued



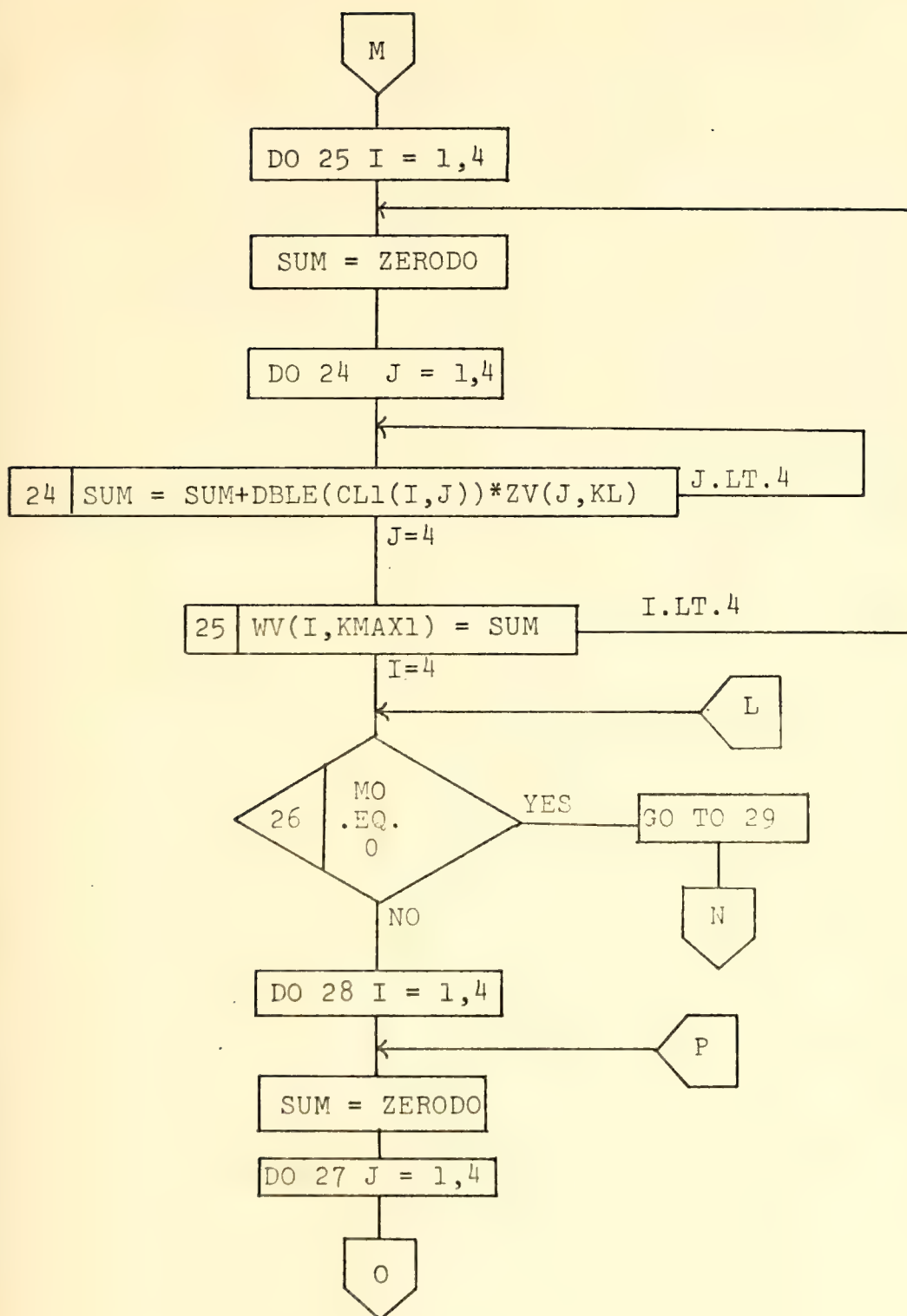


FIGURE 12 continued



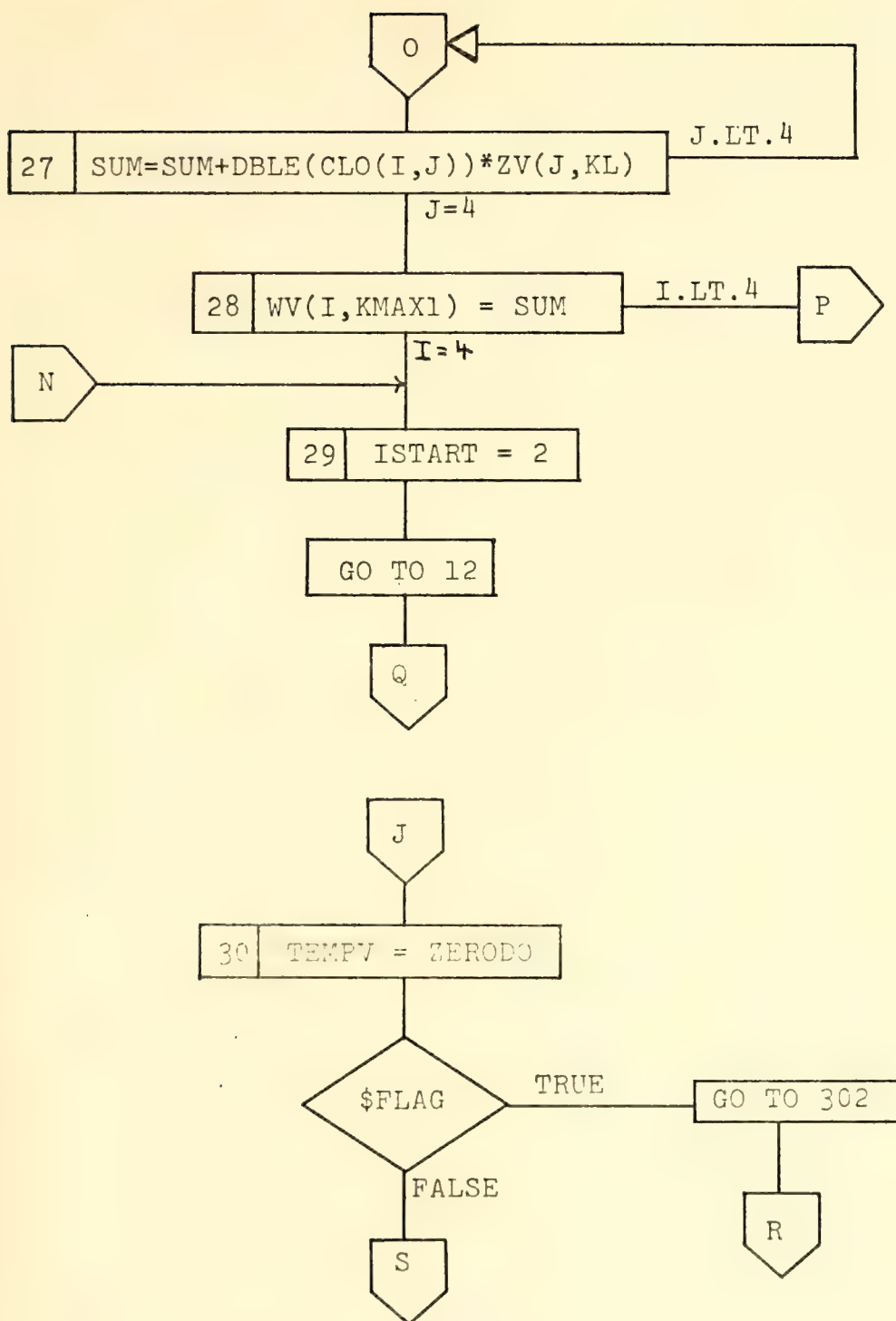


FIGURE 12 continued



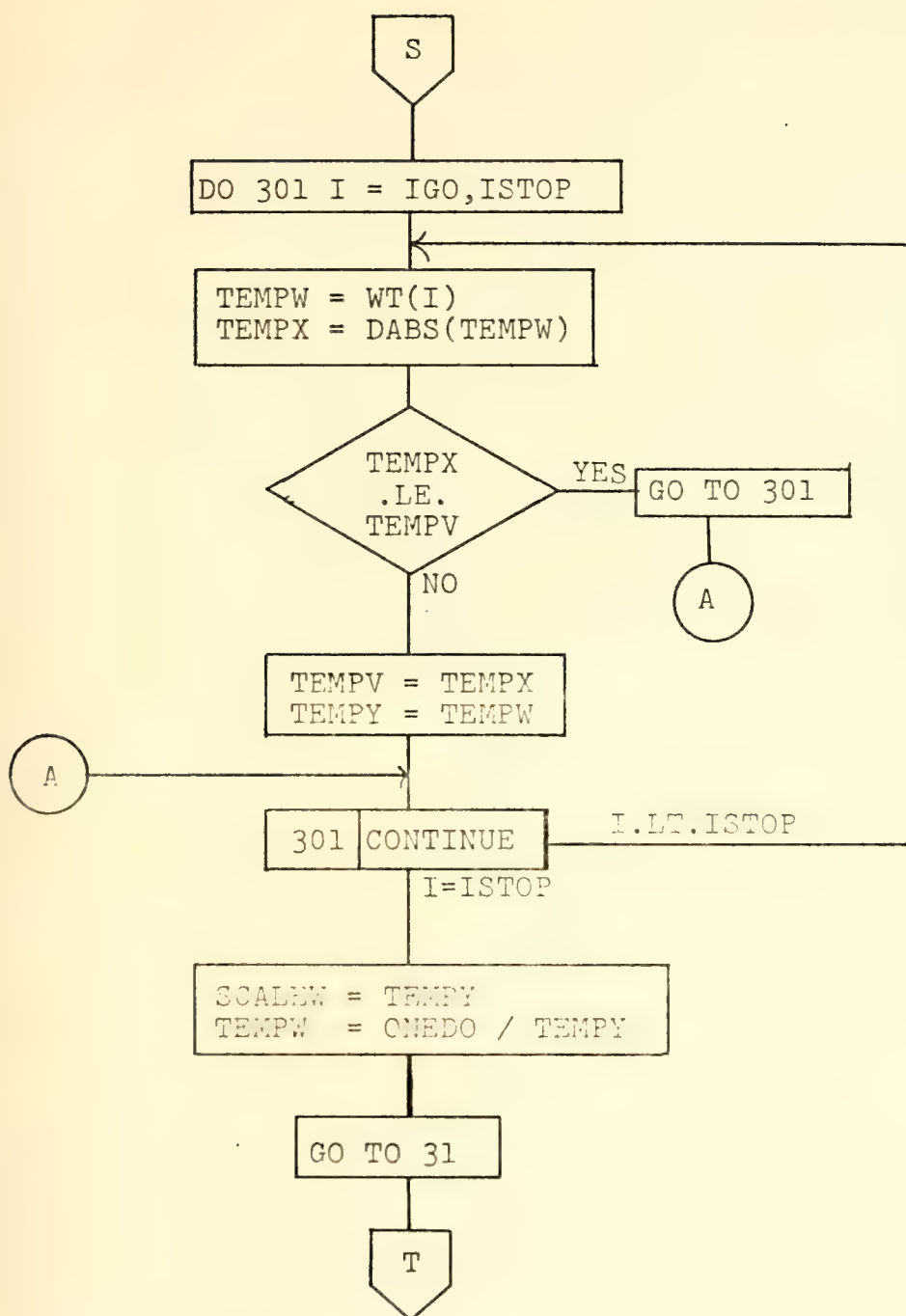


FIGURE 12 continued





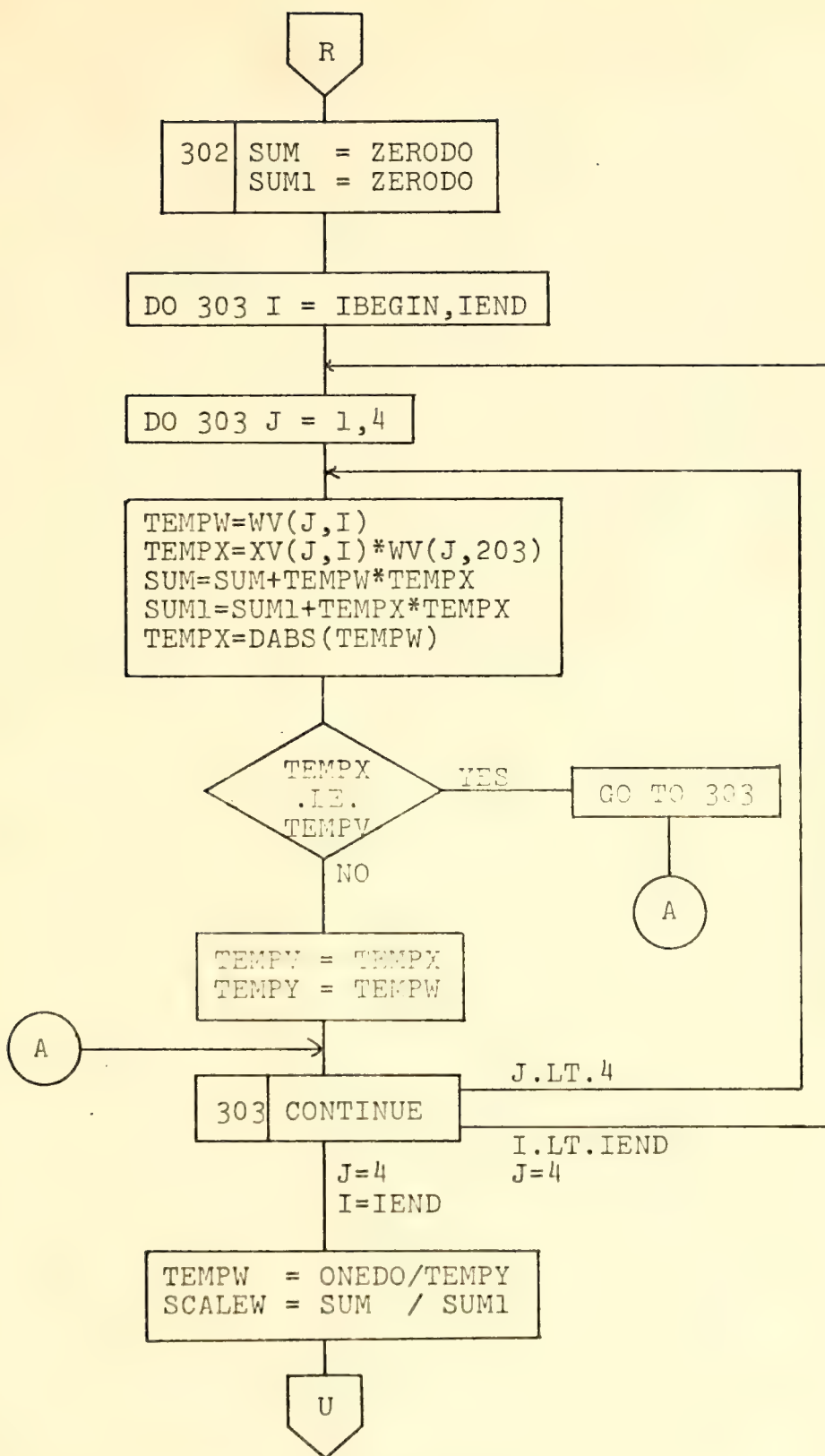


FIGURE 12 continued



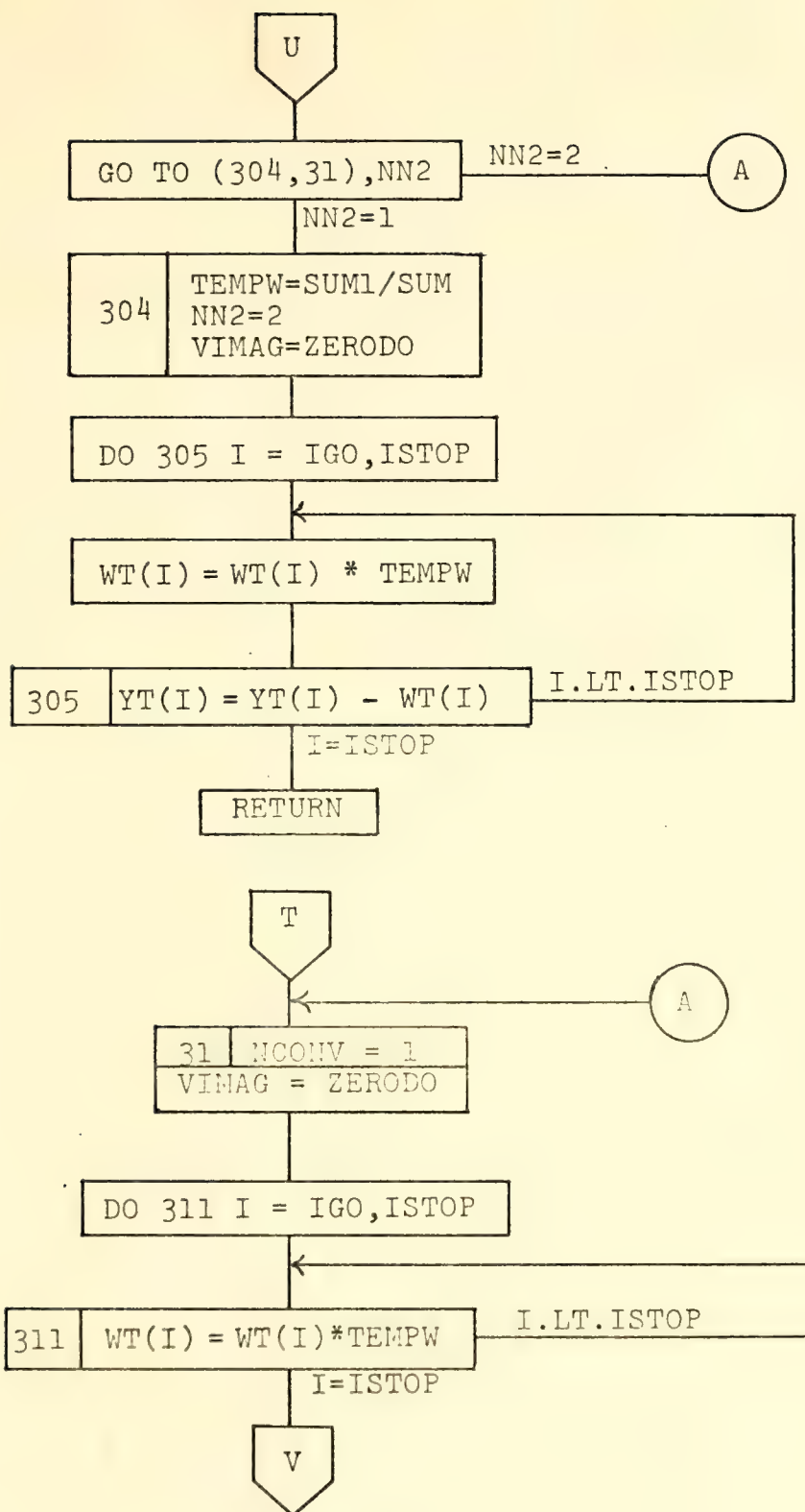


FIGURE 12 continued



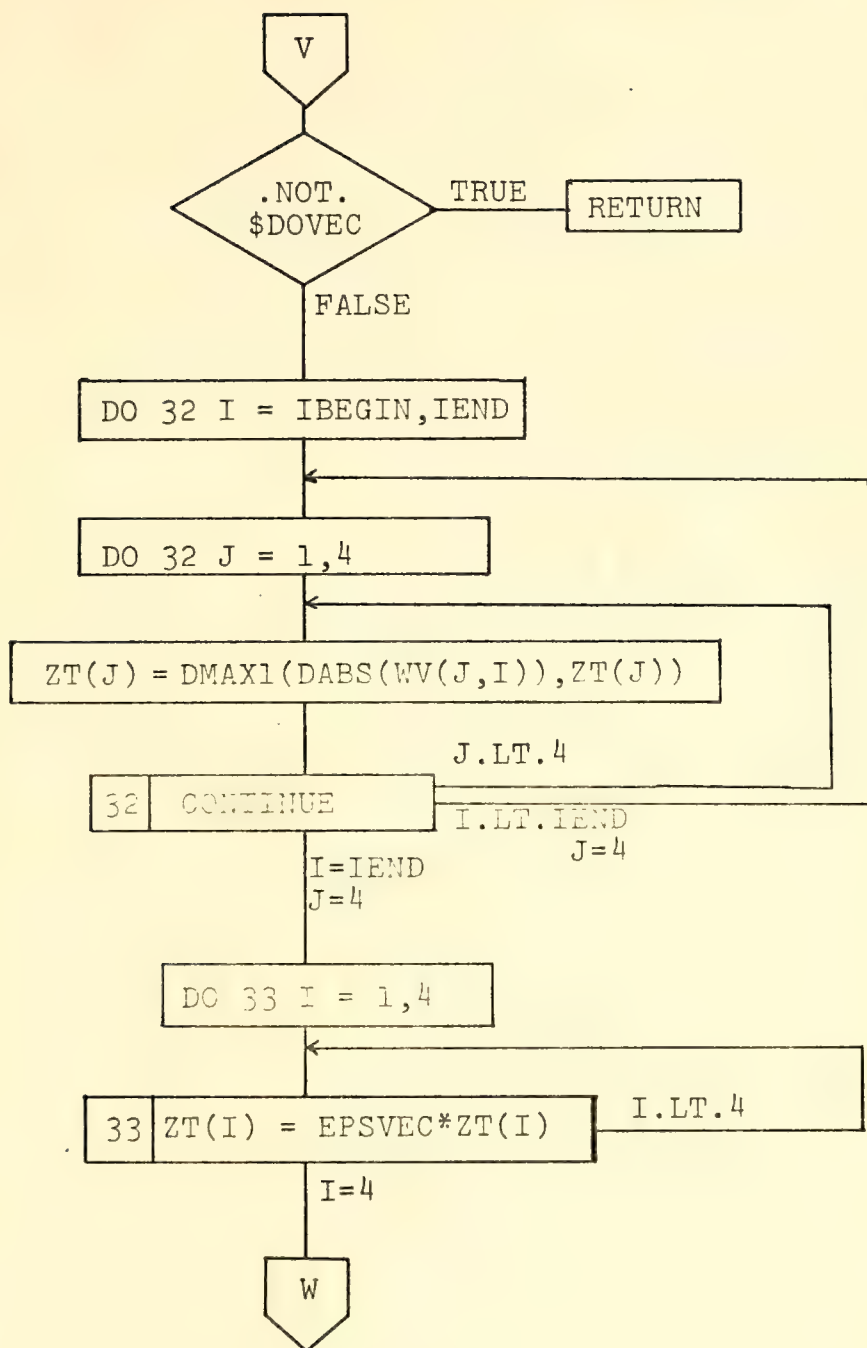


FIGURE 12 continued



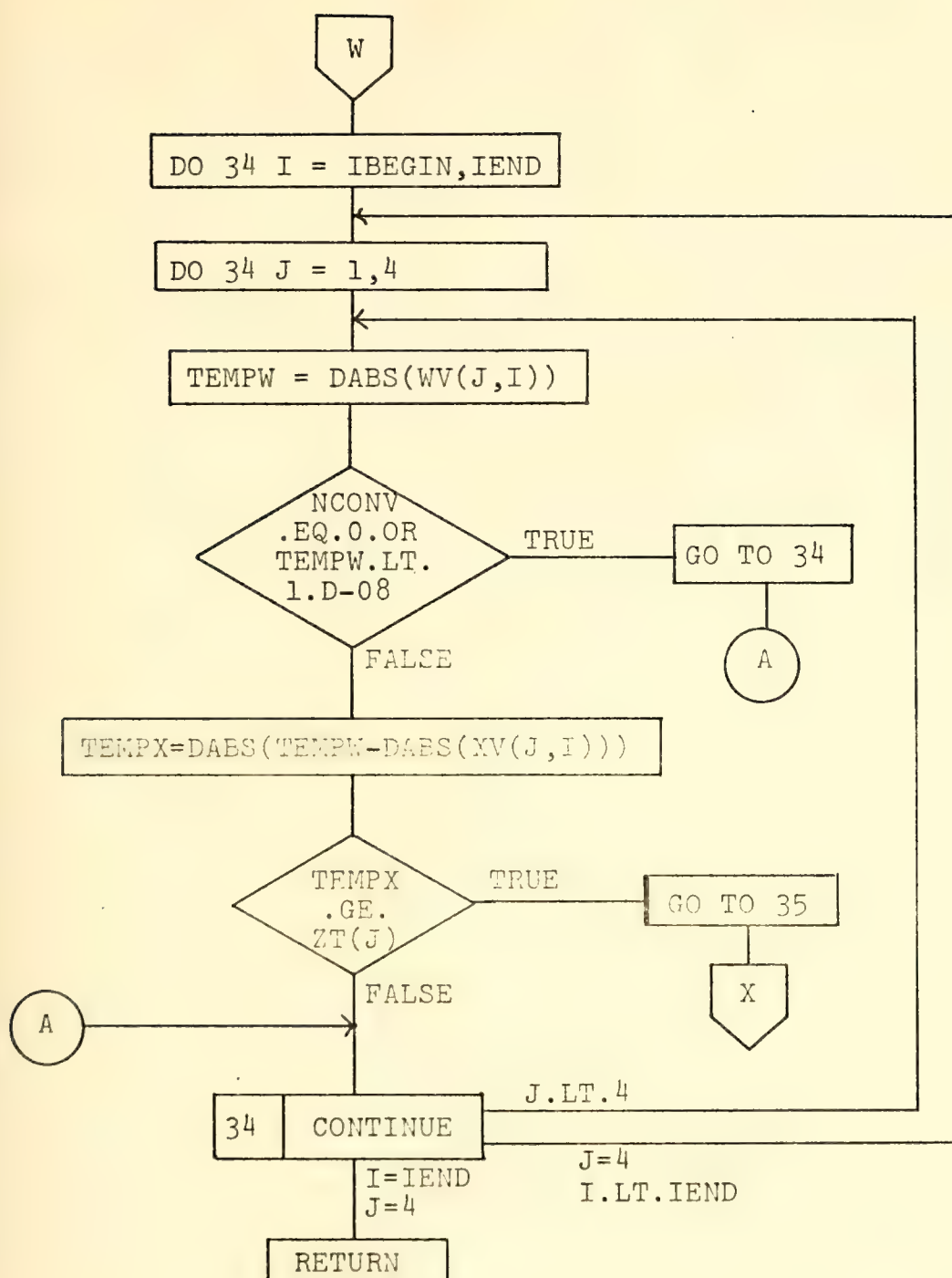


FIGURE 12 continued





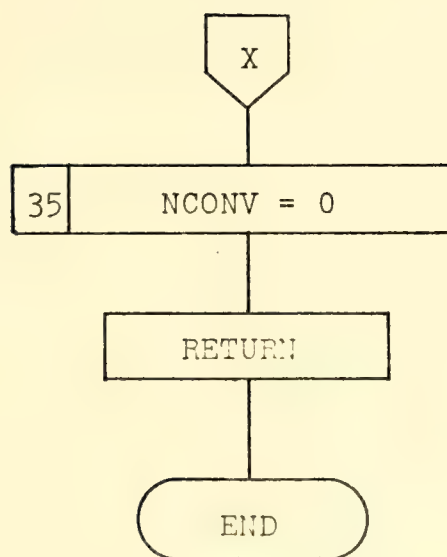


FIGURE 12 continued



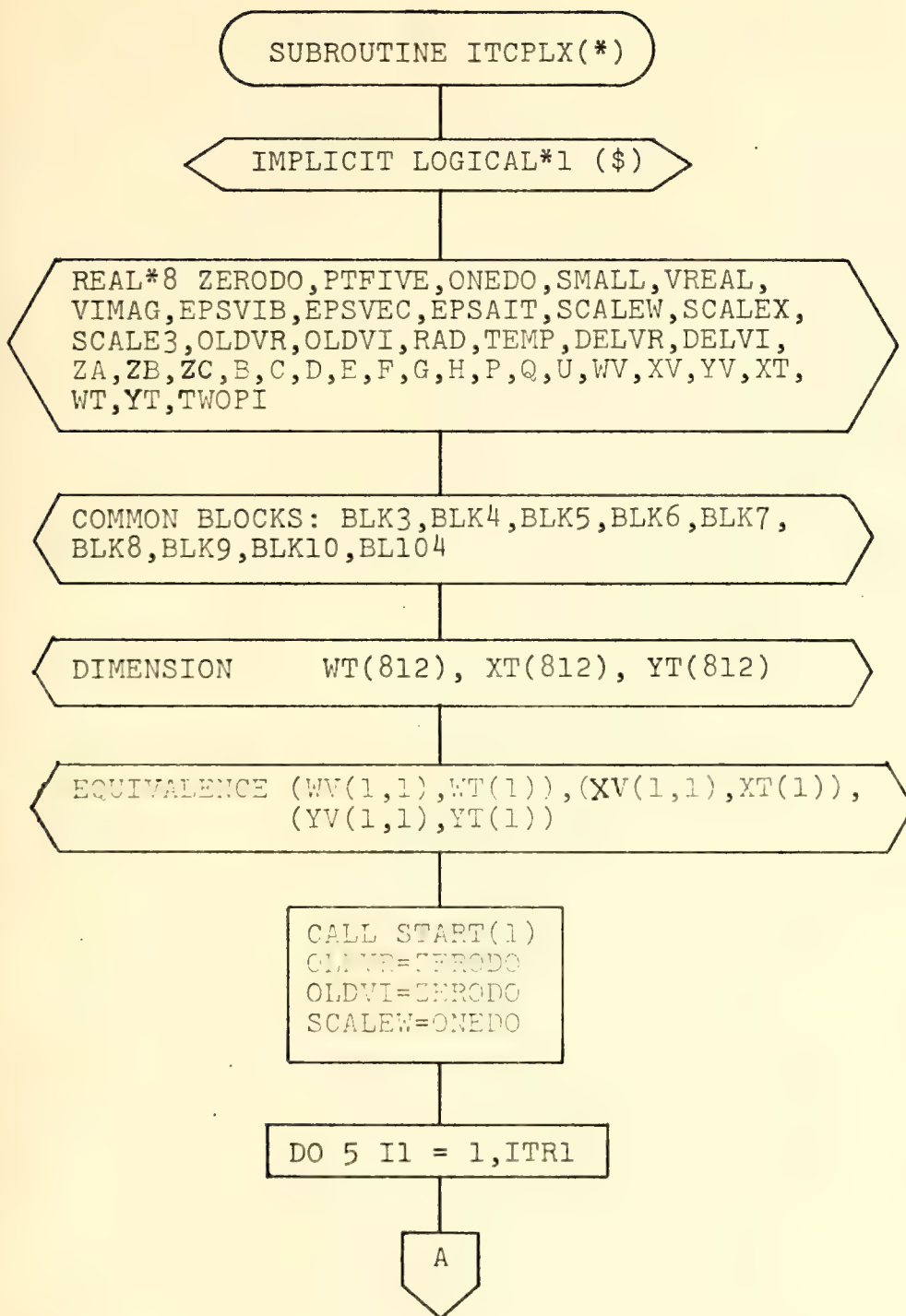


FIGURE 13 SUBROUTINE ITCPLX(\*) FLOWCHART



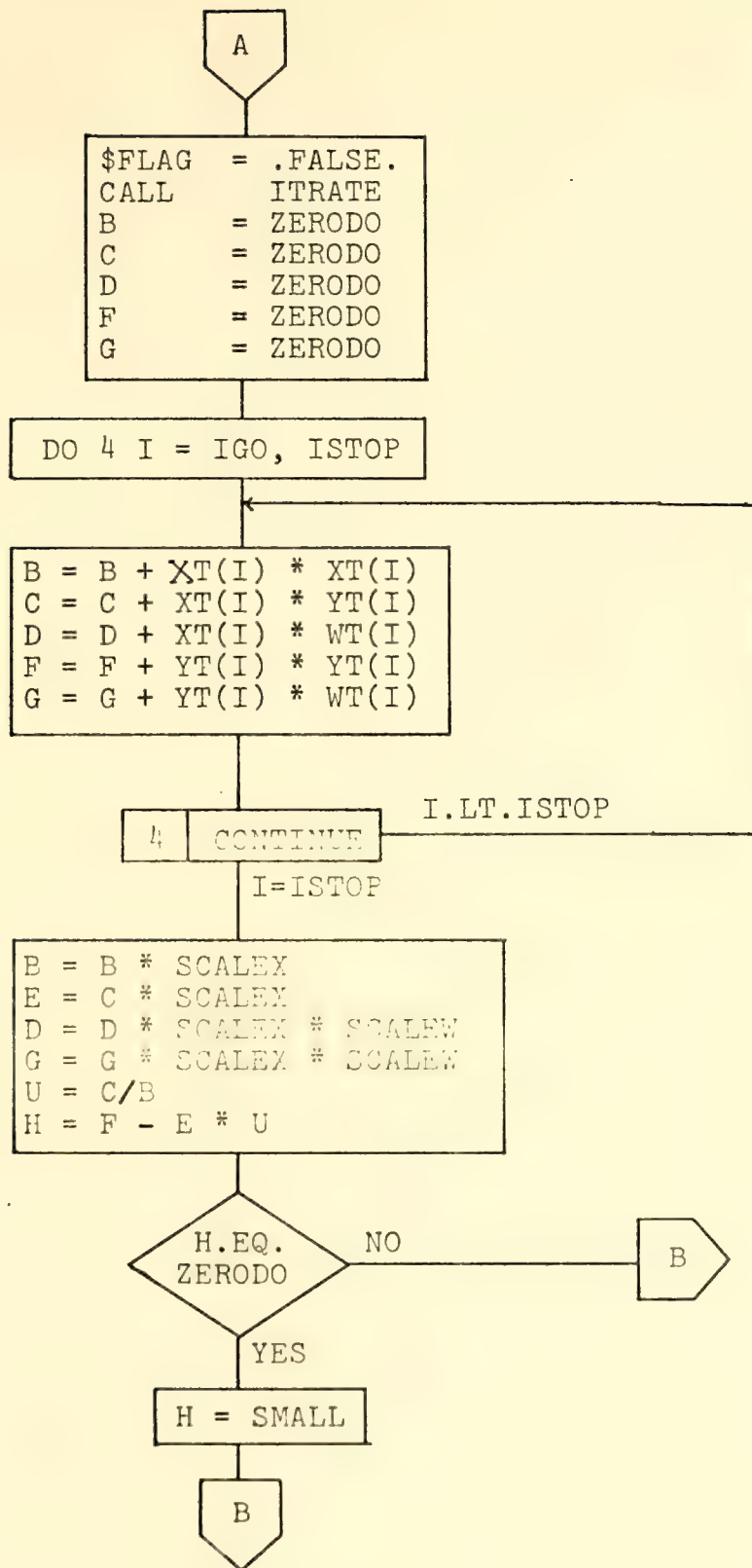


FIGURE 13 continued



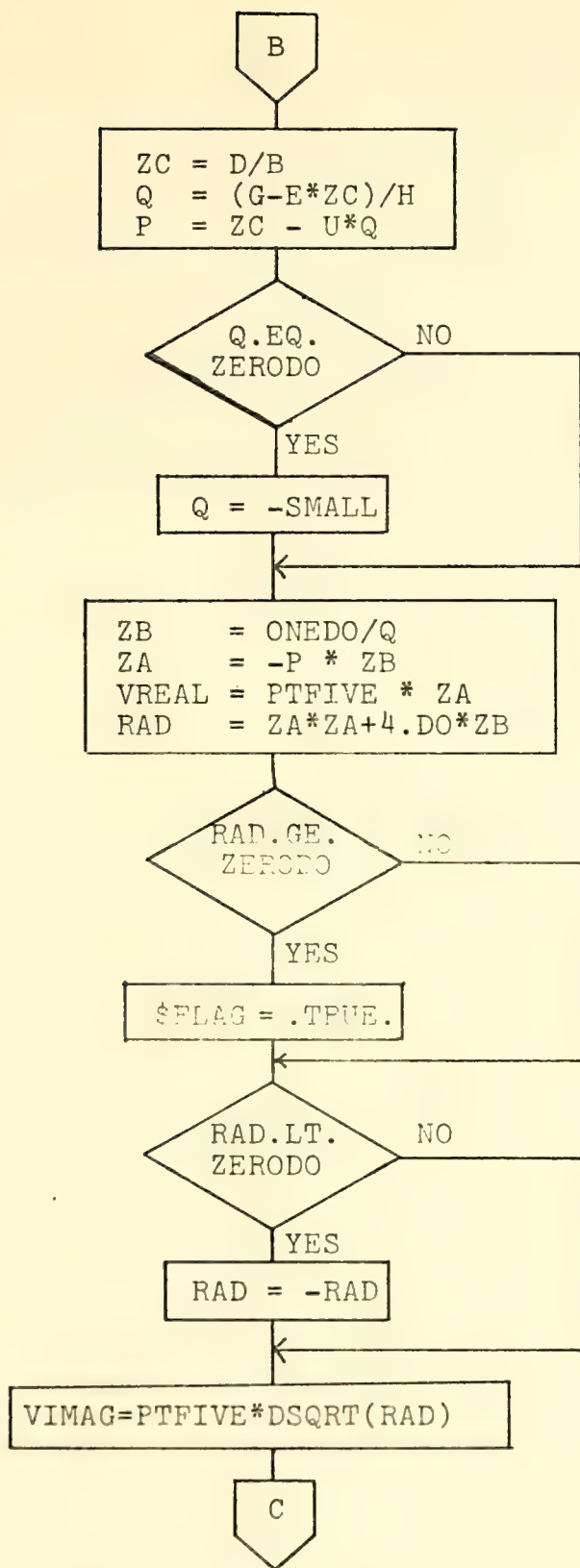


FIGURE 13 continued





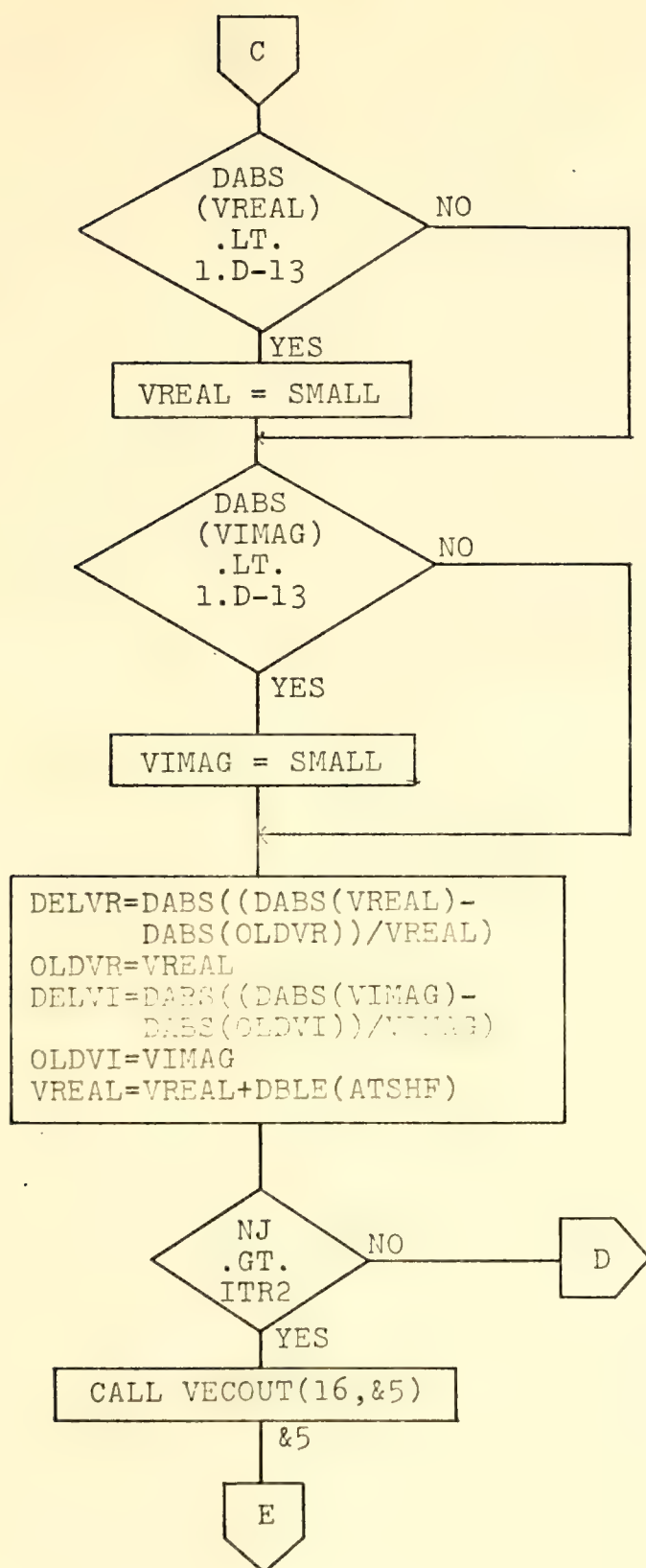


FIGURE 13 continued



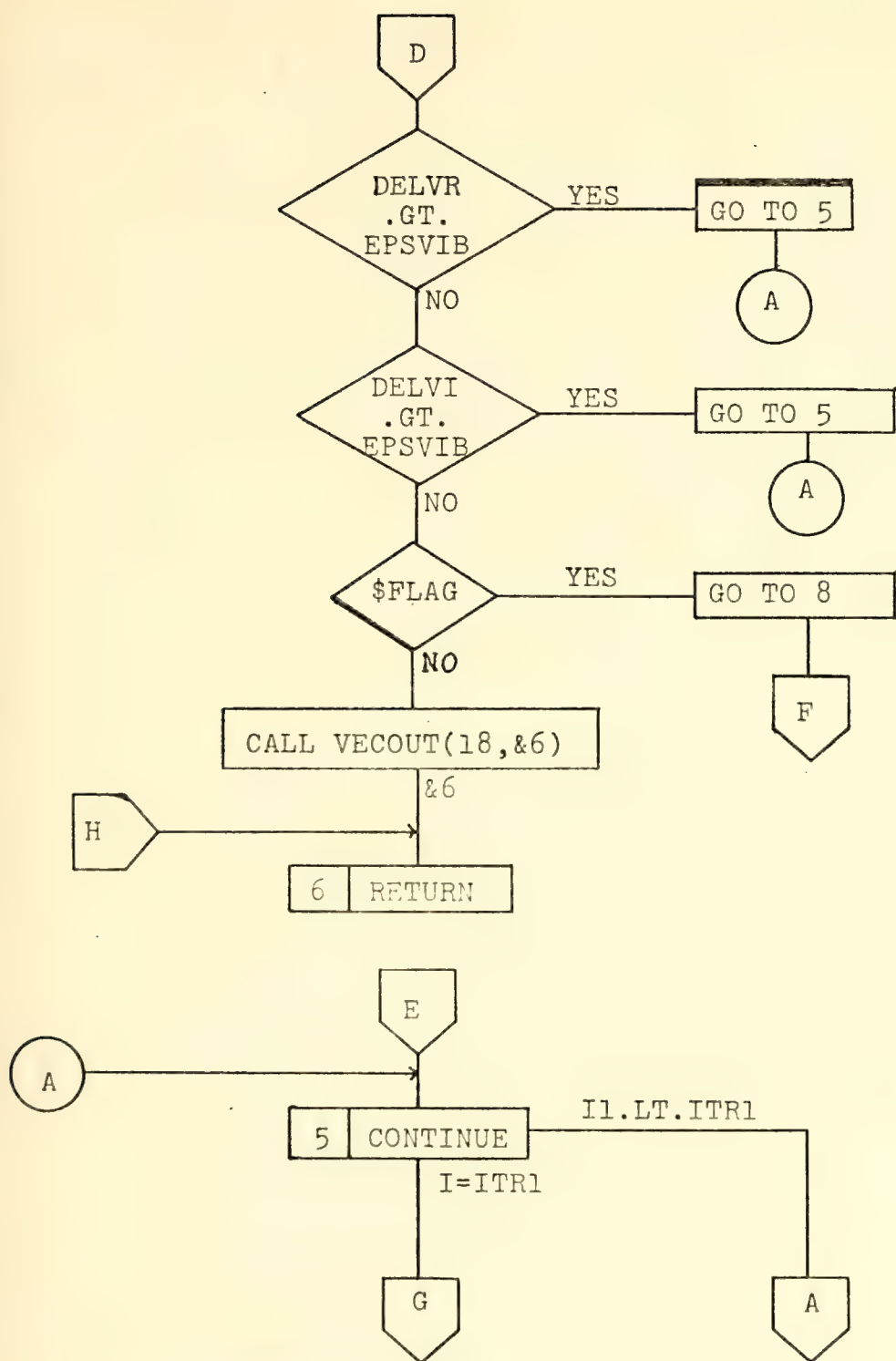


FIGURE 13 continued



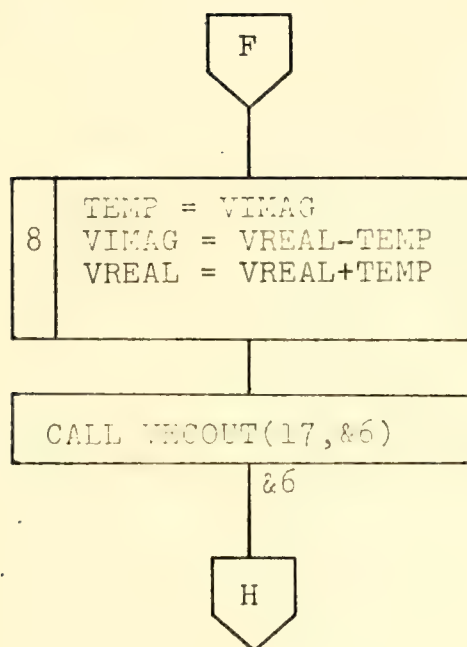
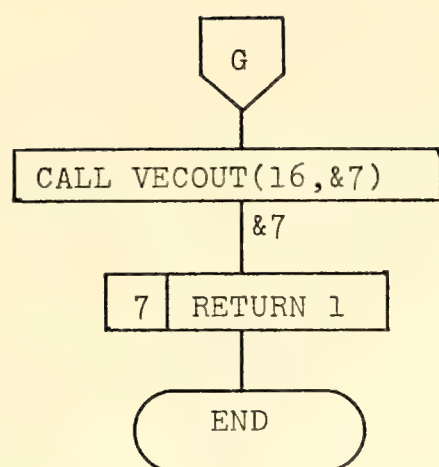


FIGURE 13 continued



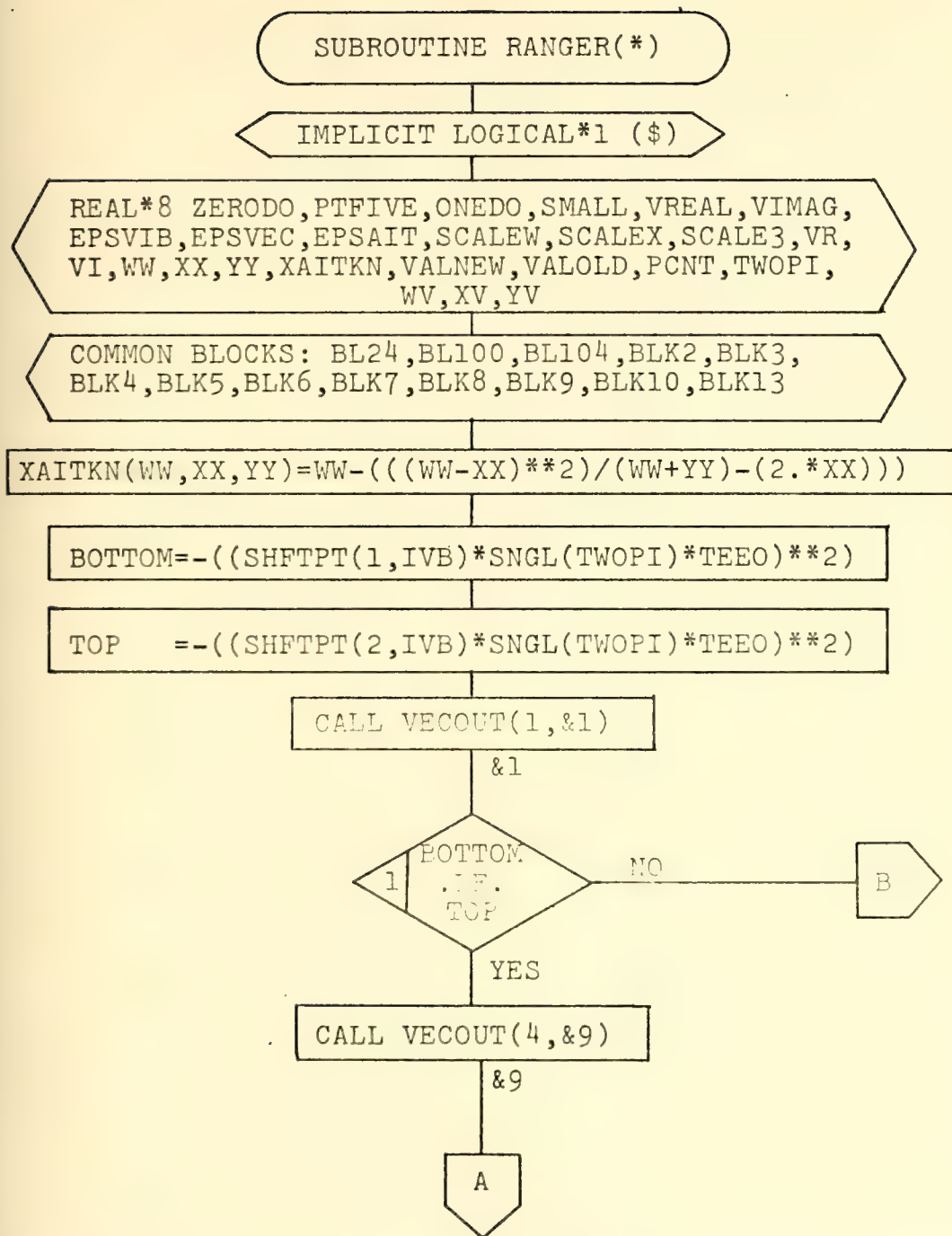


FIGURE 14. SUBROUTINE RANGER(\*) FLOWCHART





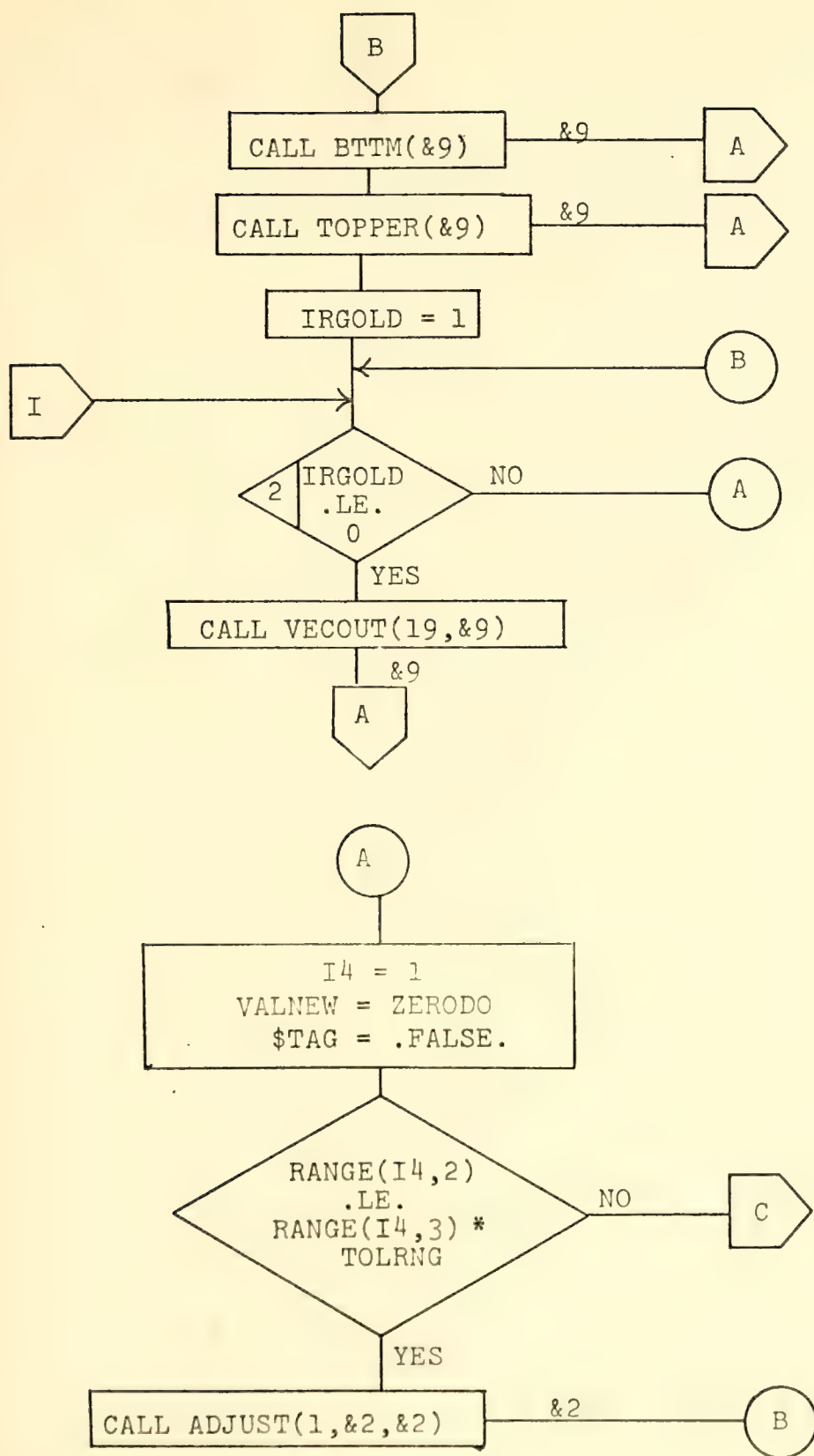


FIGURE 14 continued



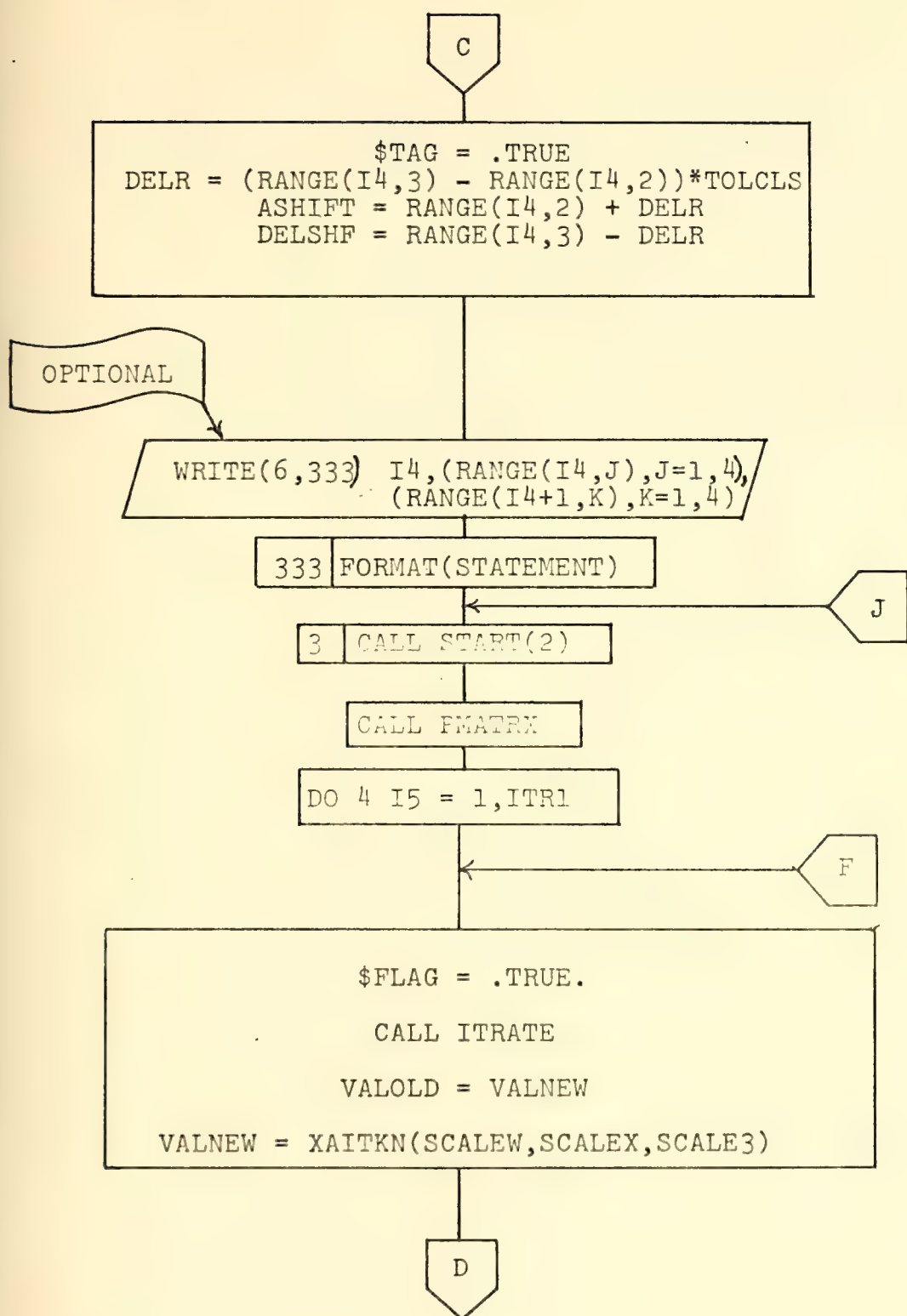


FIGURE 14 continued



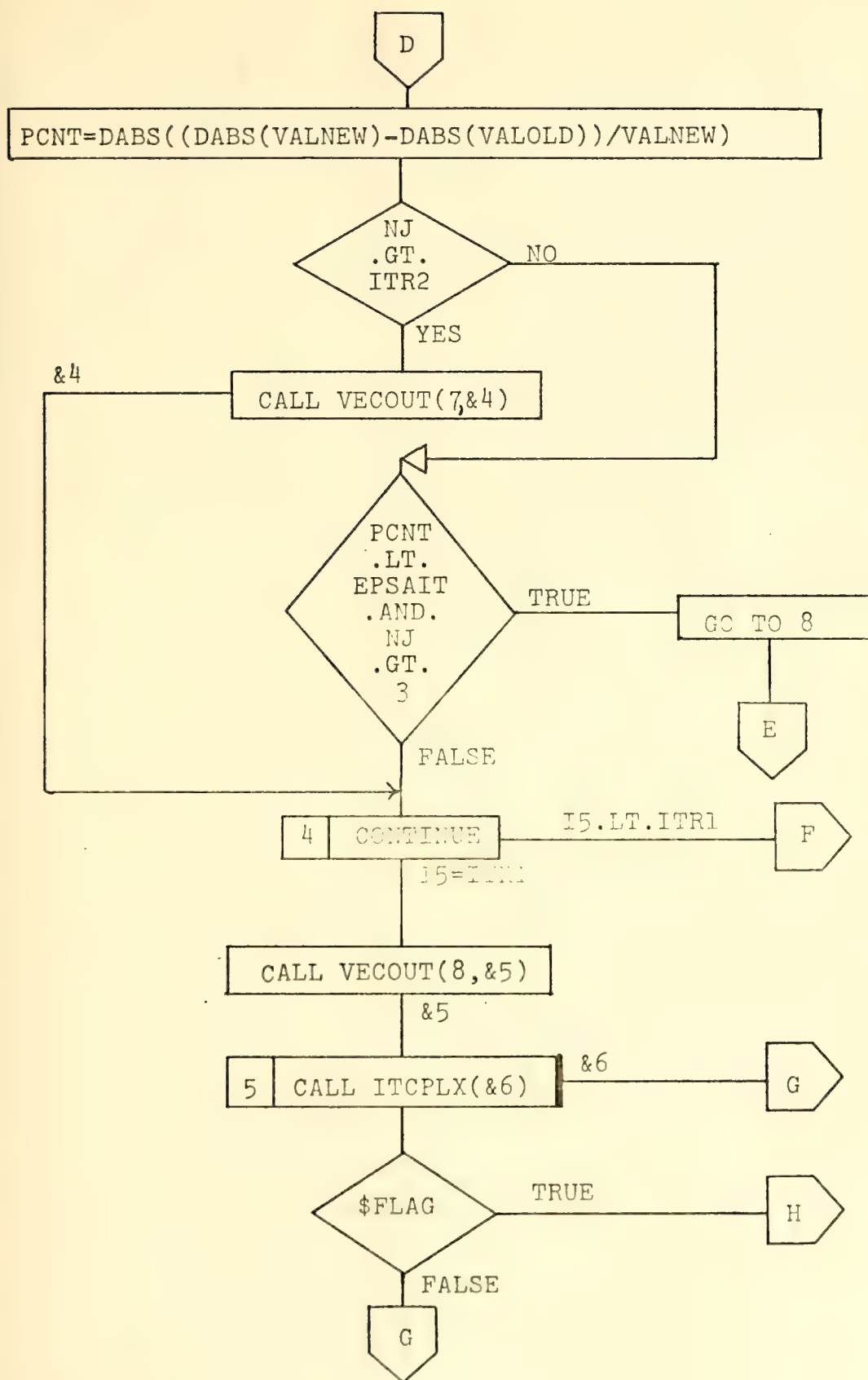


FIGURE 14 continued



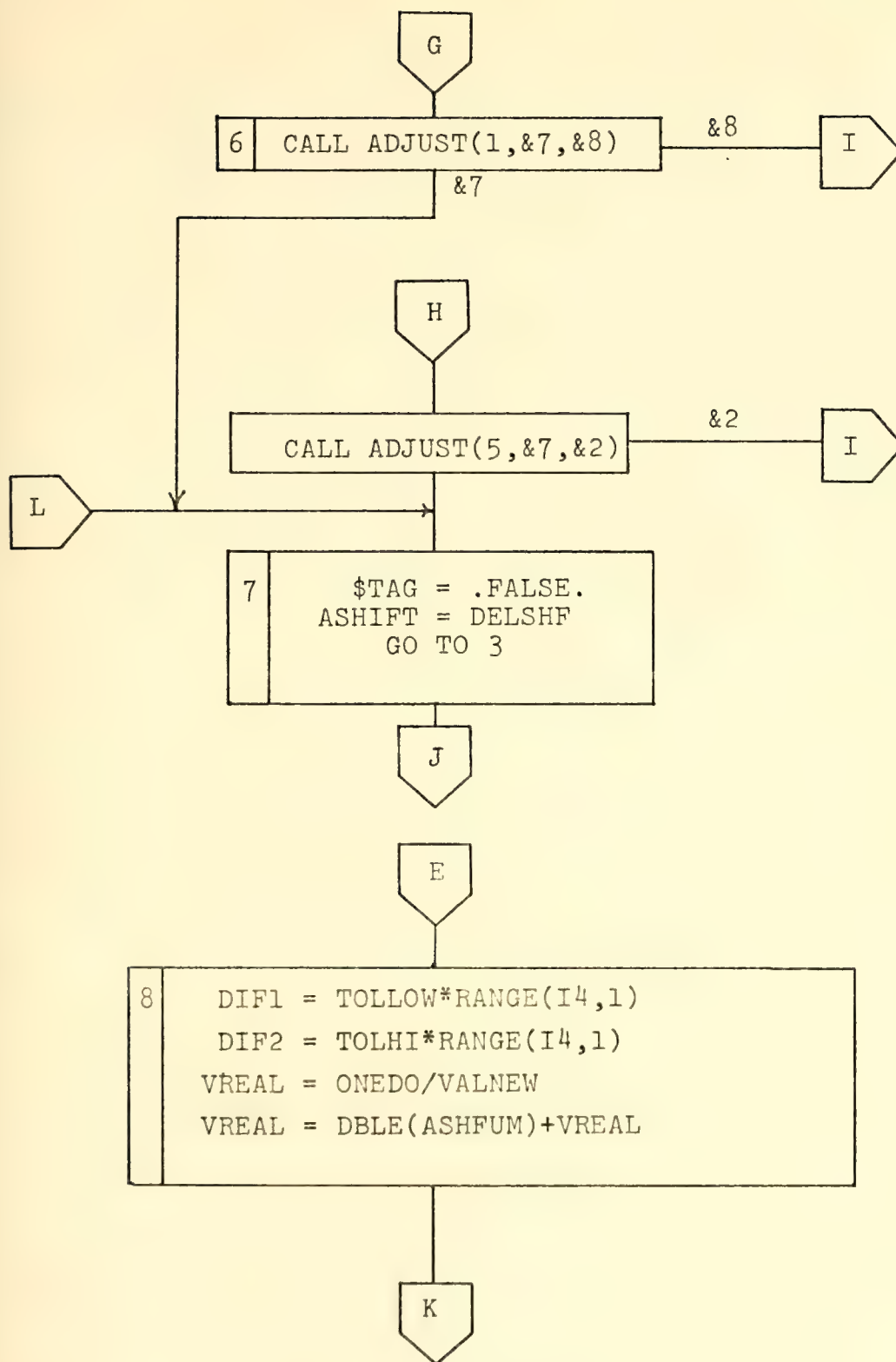


FIGURE 14 continued





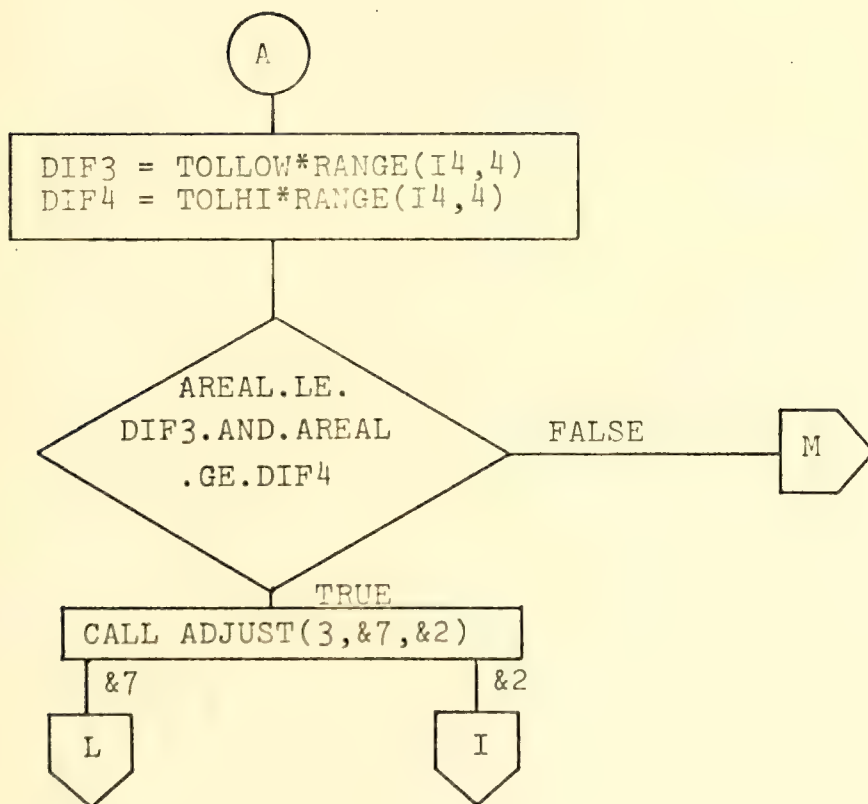
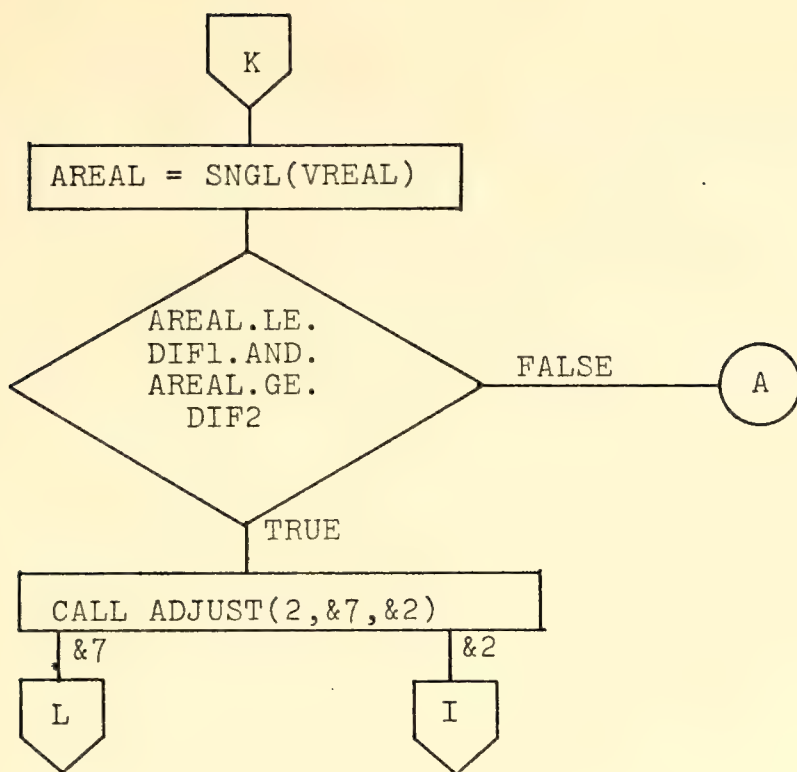


FIGURE 14 continued



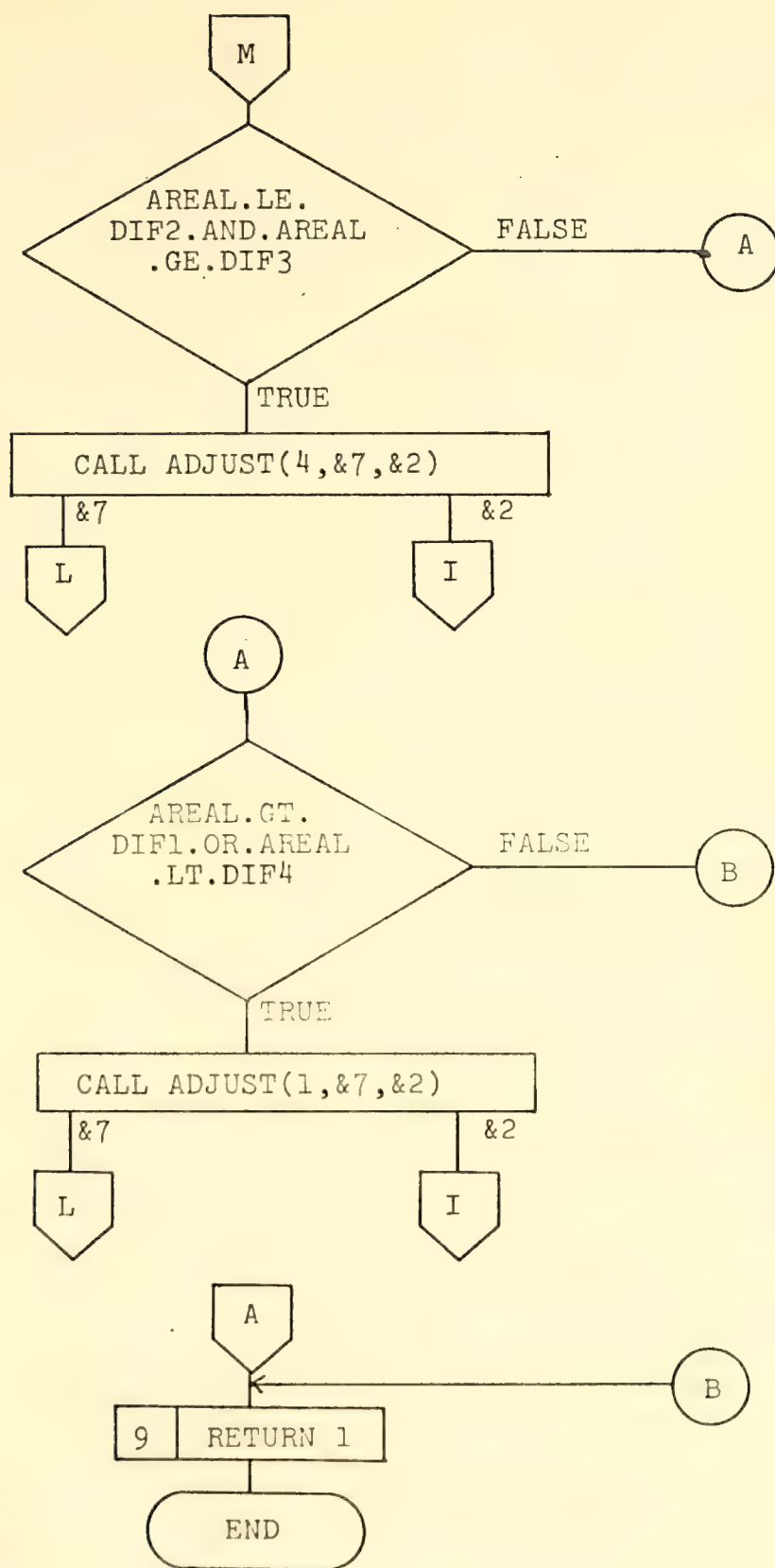


FIGURE 14 continued



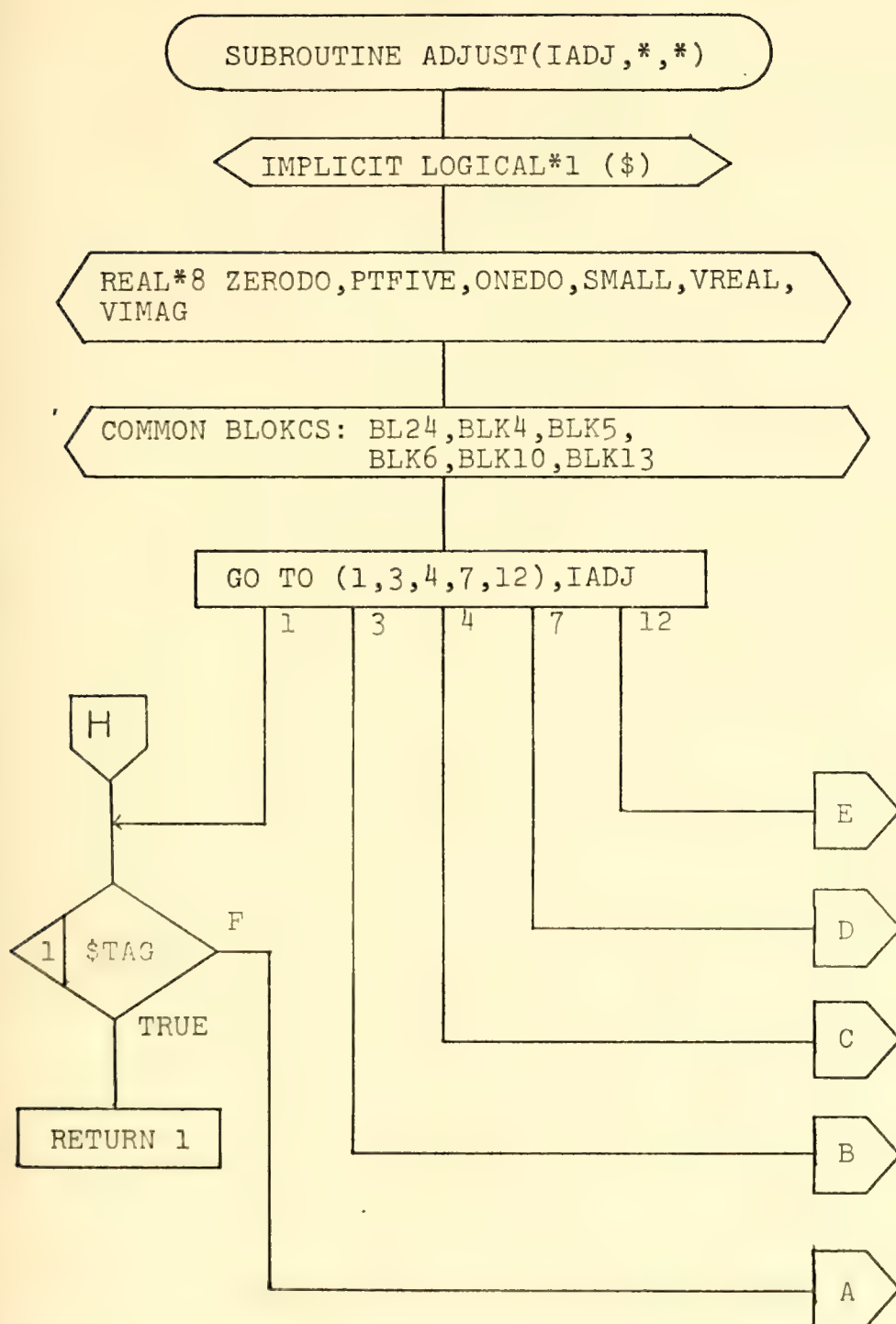


FIGURE 15. SUBROUTINE ADJUST(IADJ,\*,\*) FLOWCHART



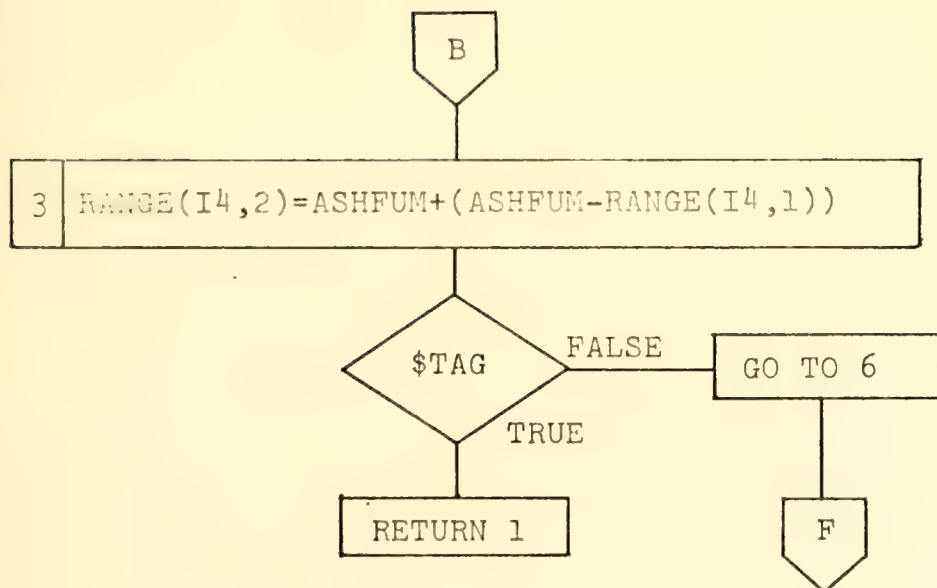
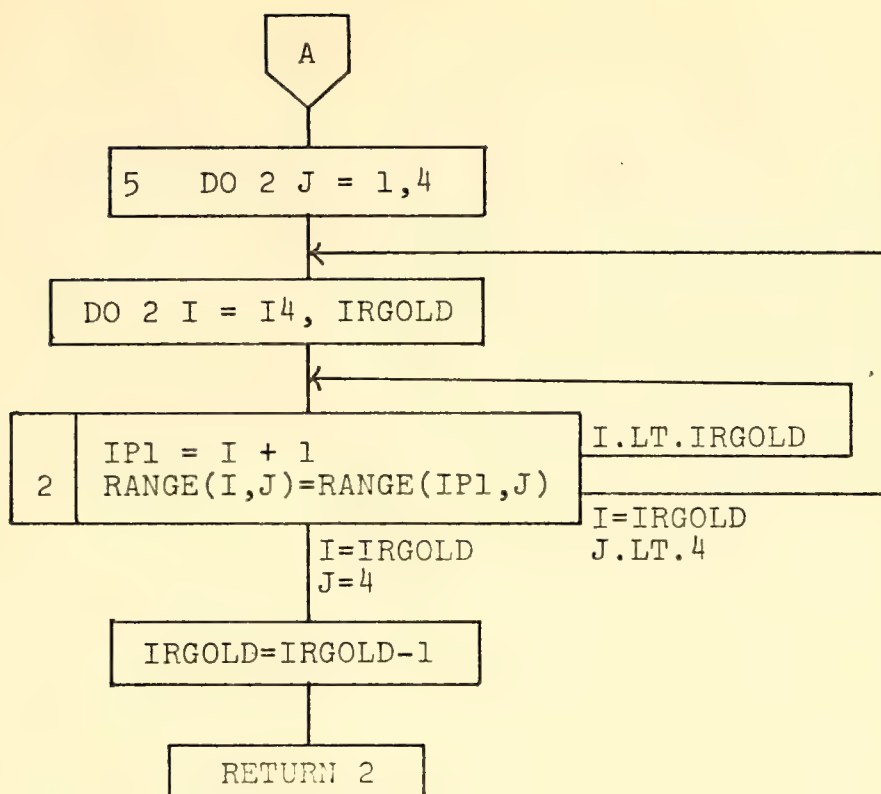


FIGURE 15 continued





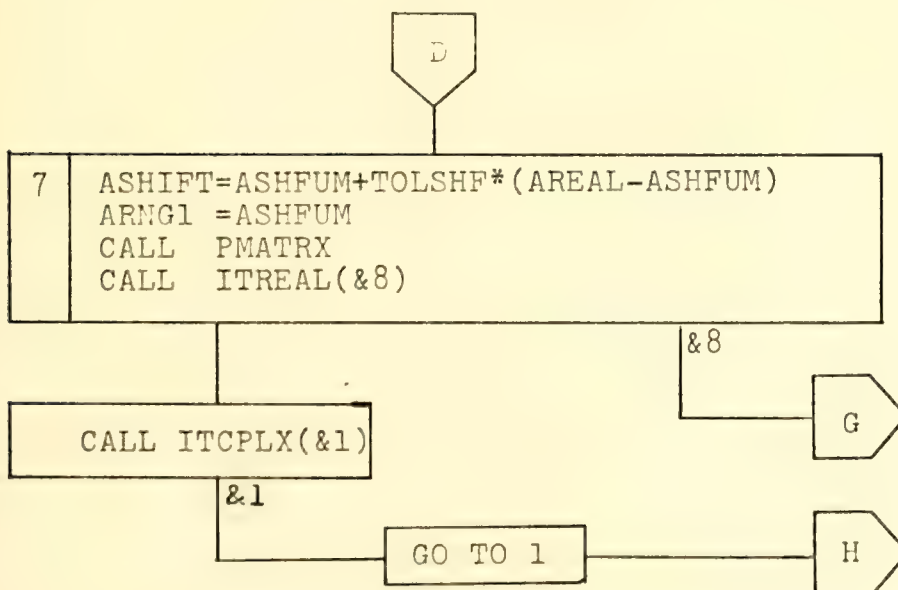
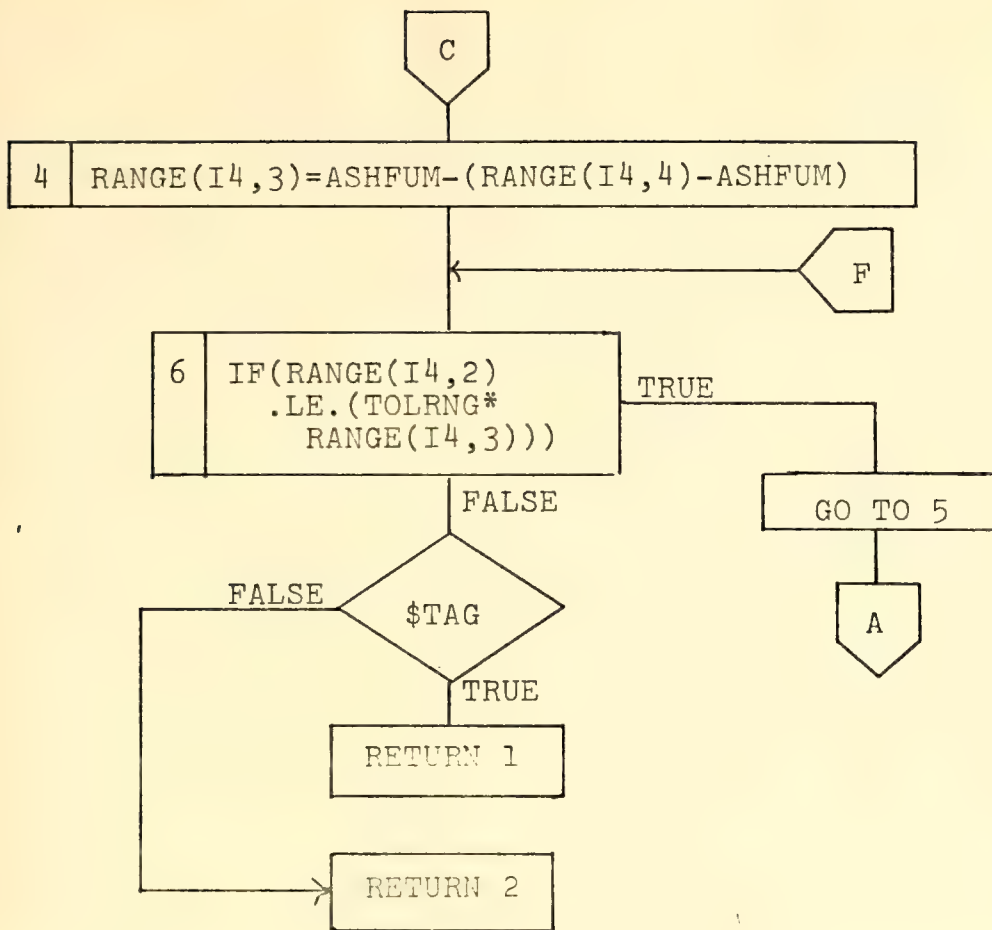


FIGURE 15 continued



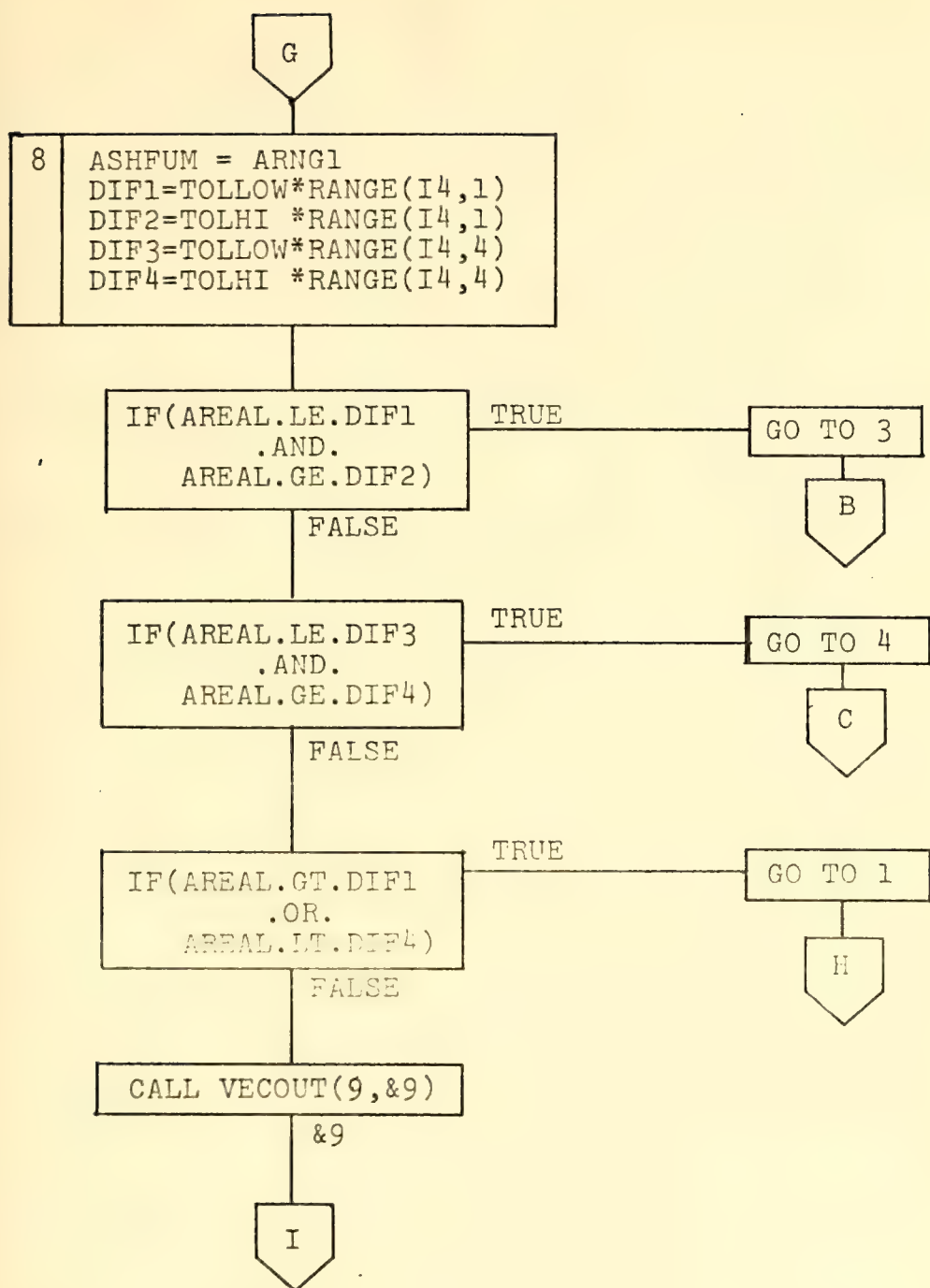


FIGURE 15 continued



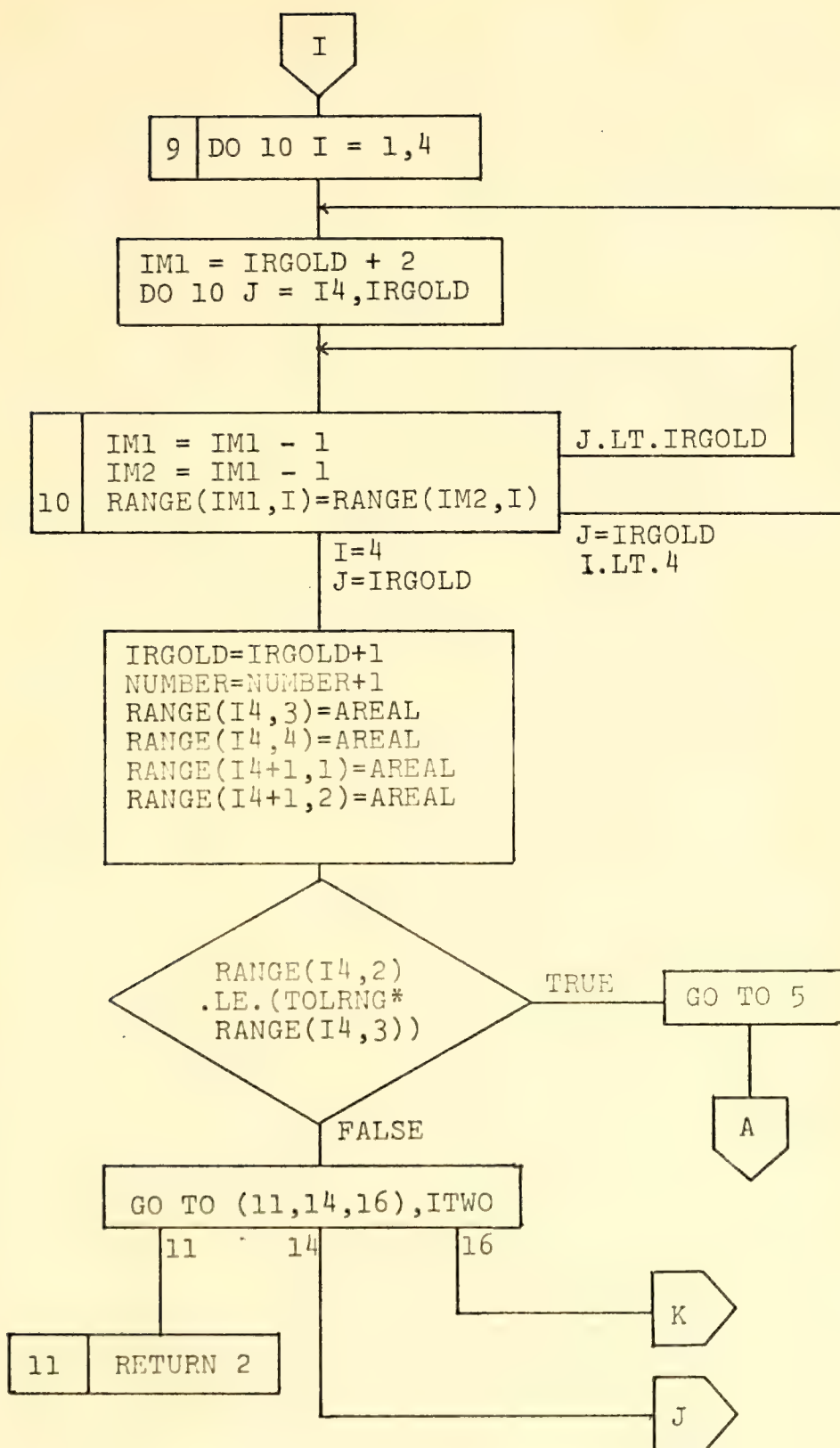


FIGURE 15 continued



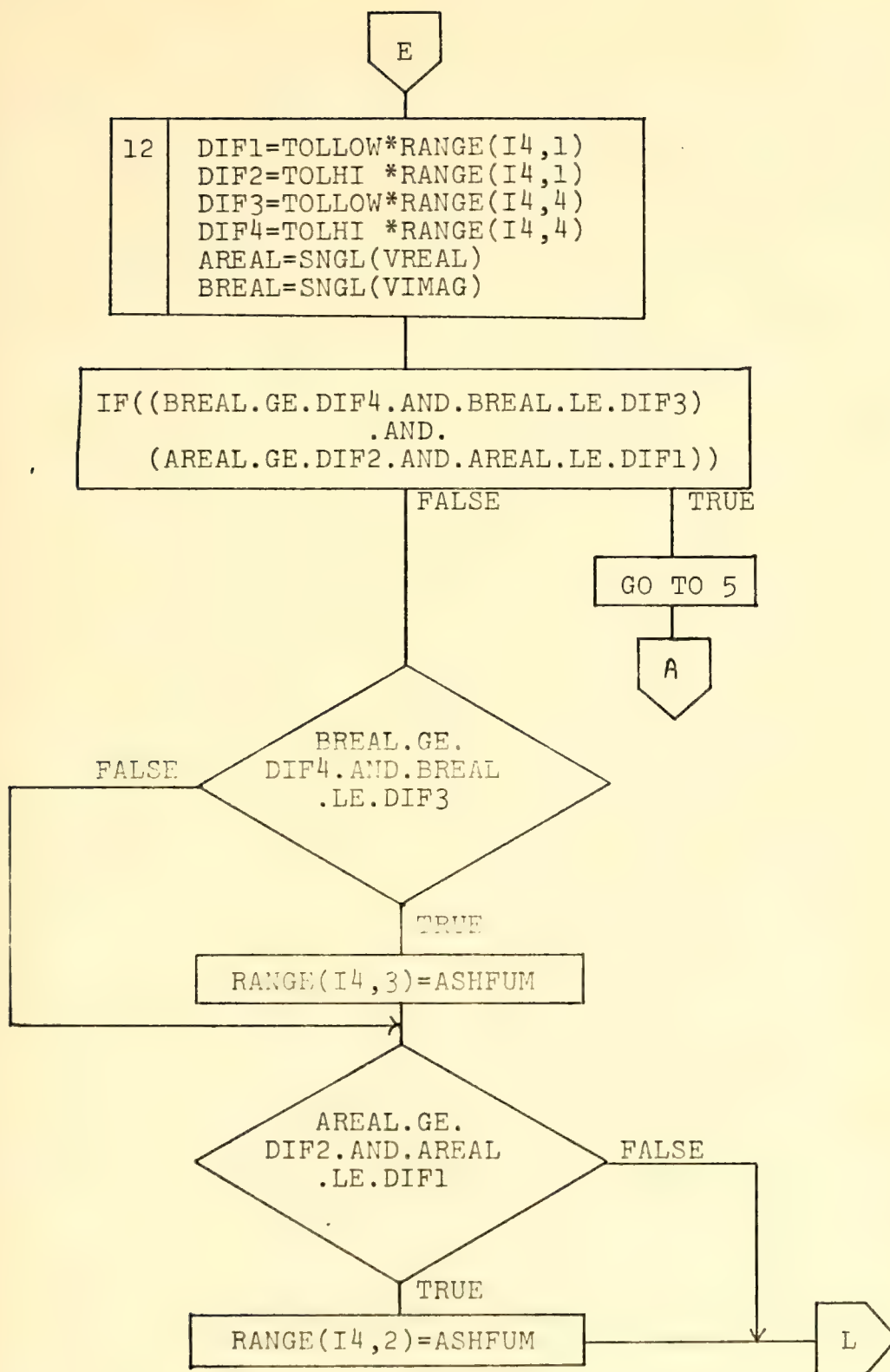


FIGURE 15 continued





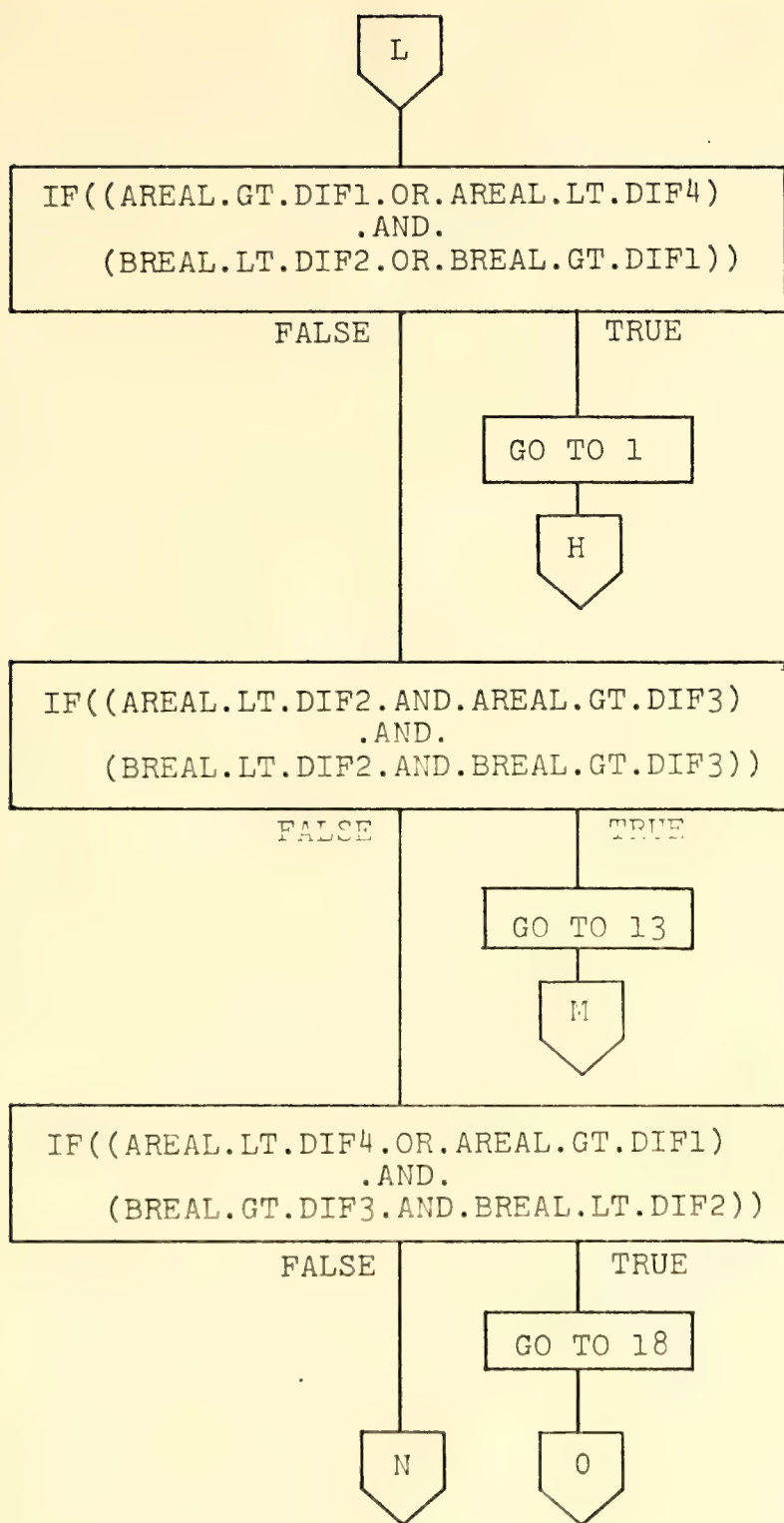


FIGURE 15 continued



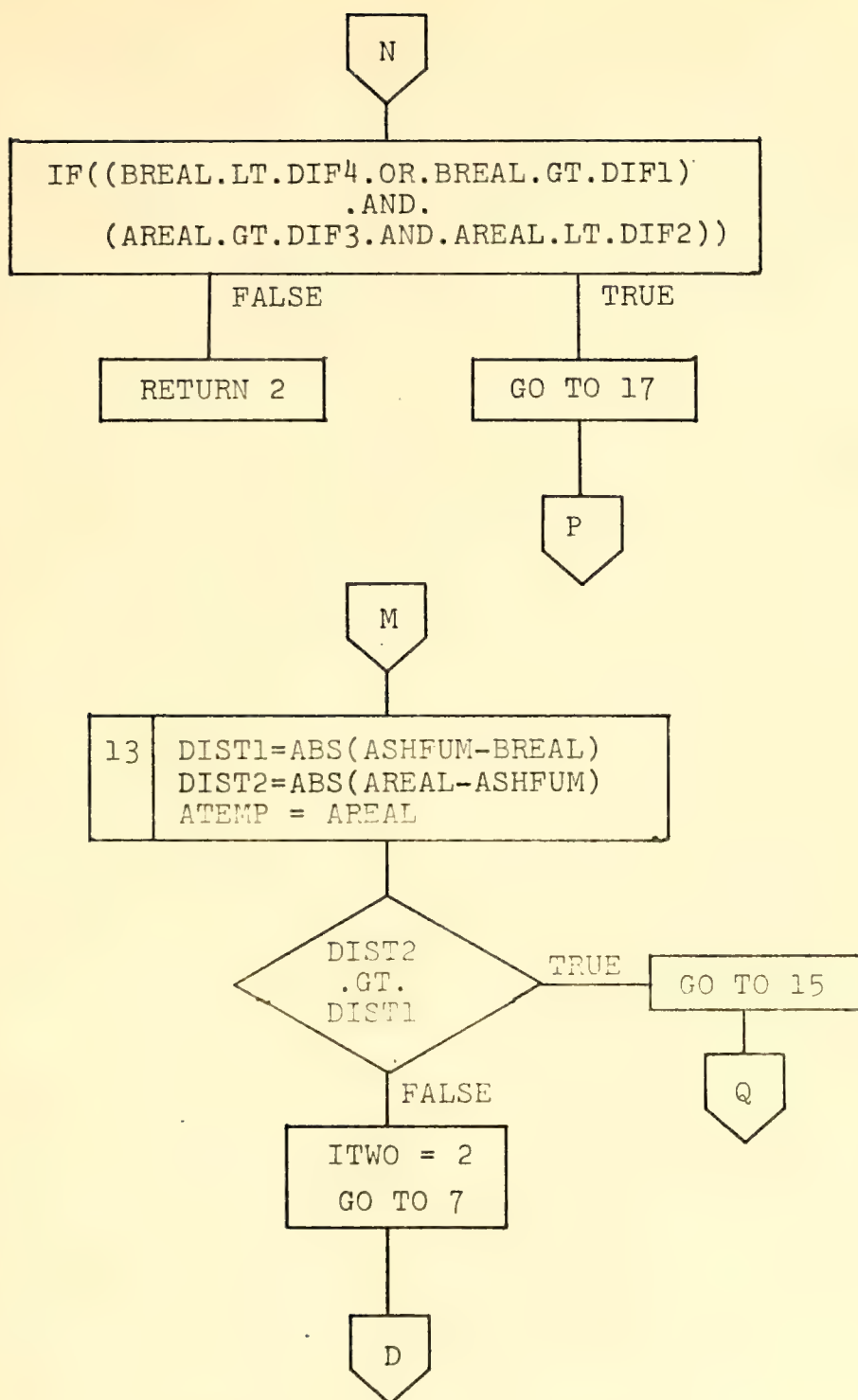


FIGURE 15 continued



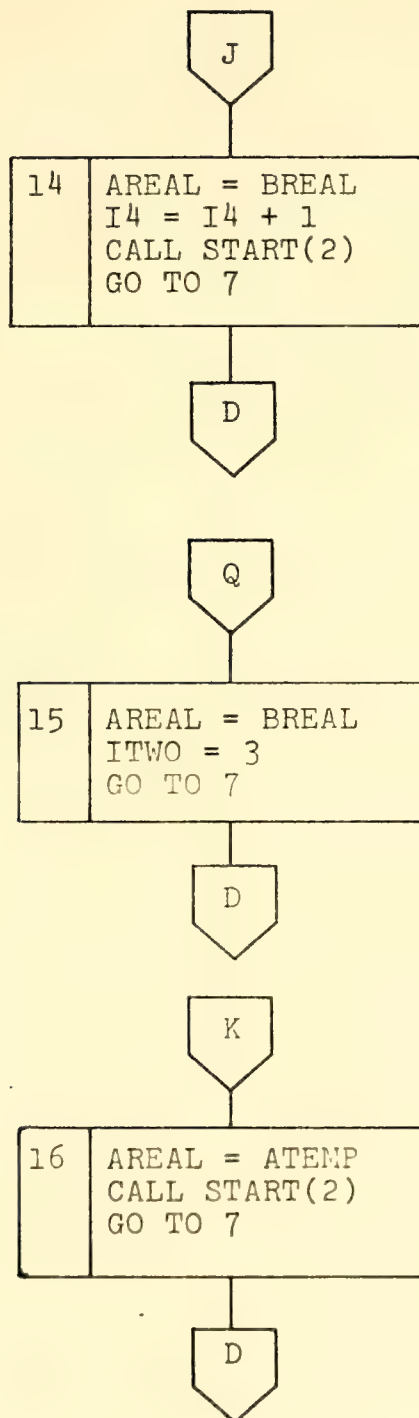


FIGURE 15 continued



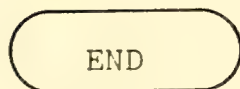
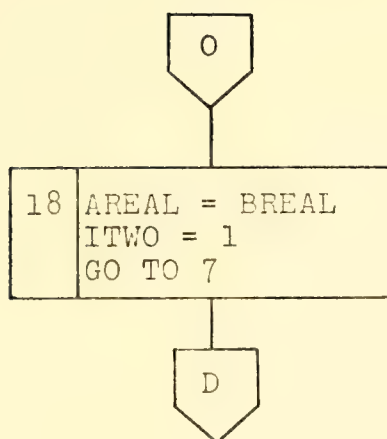
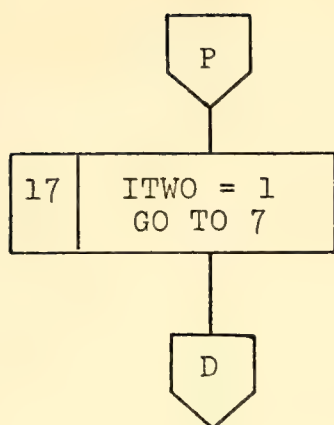


FIGURE 15 continued





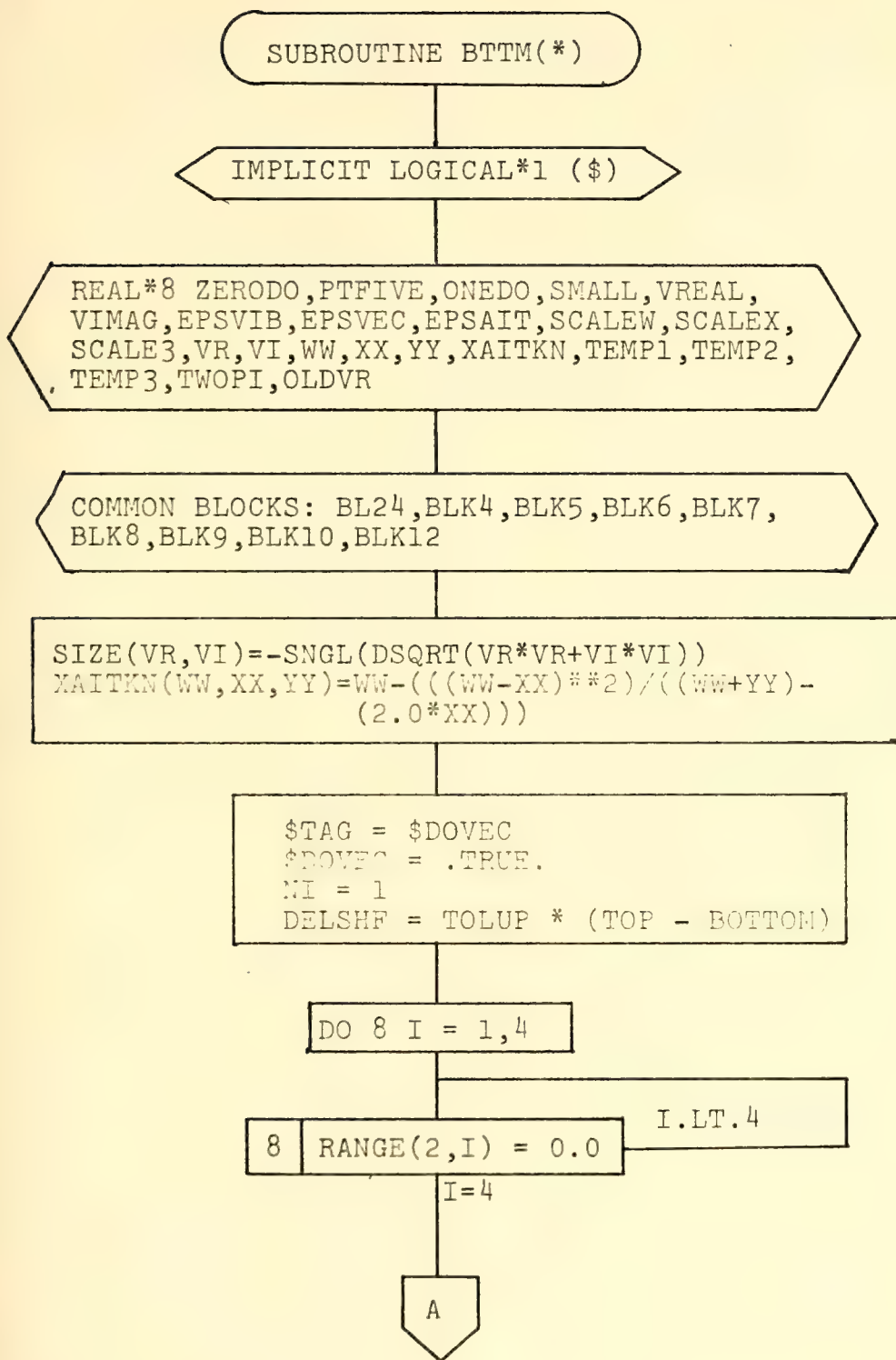


FIGURE 16 SUBROUTINE BTM(\*) FLOWCHART



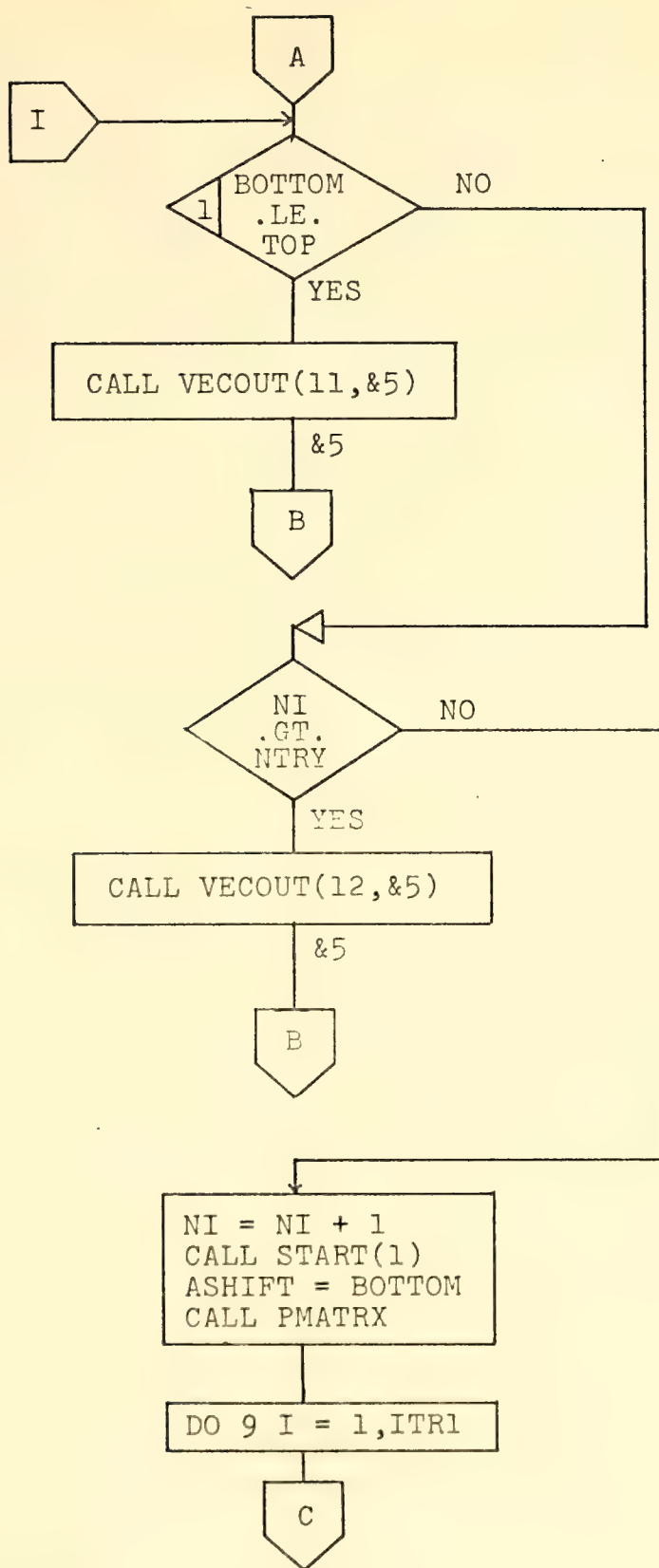


FIGURE 16 continued



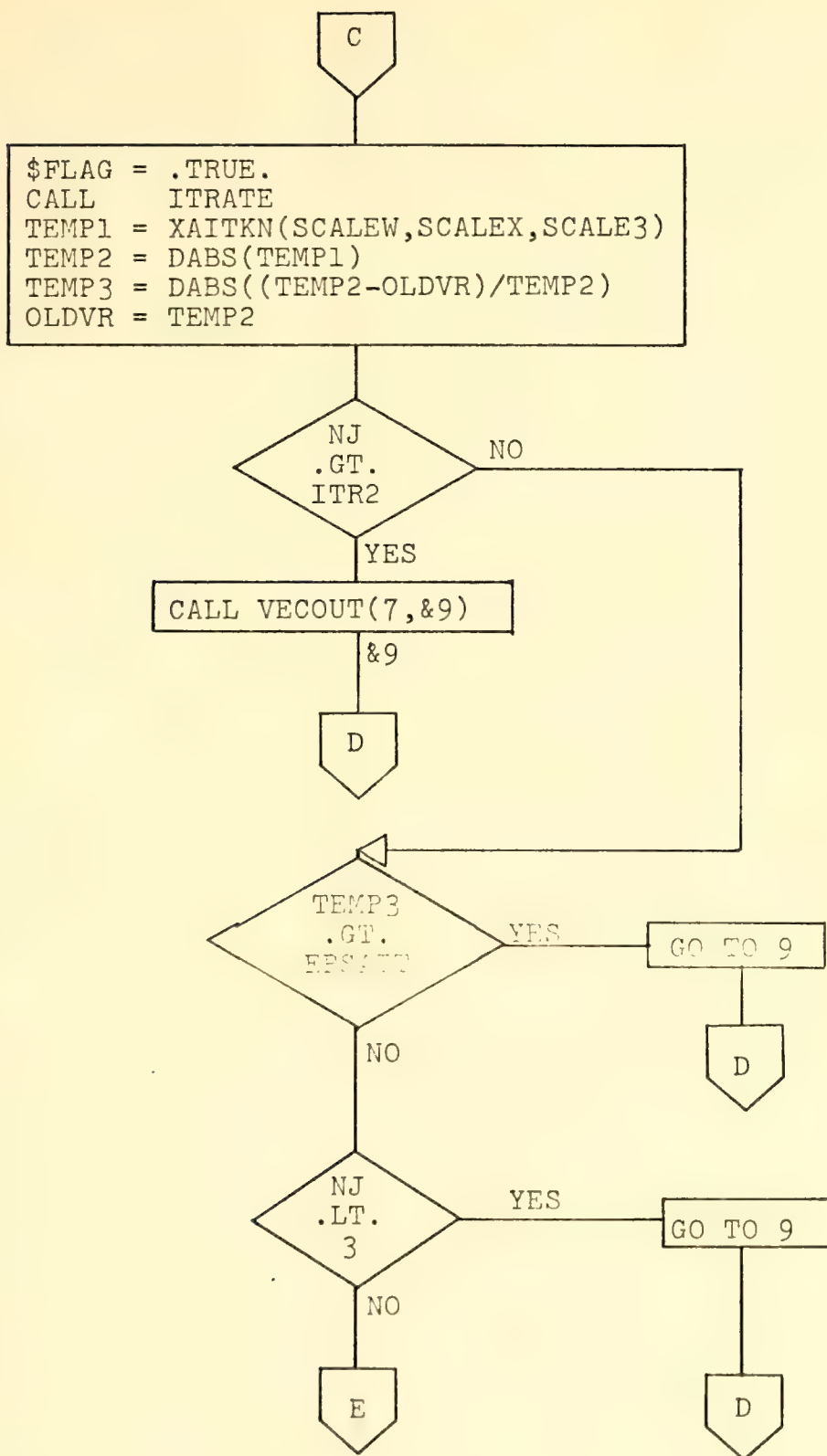


FIGURE 16 continued



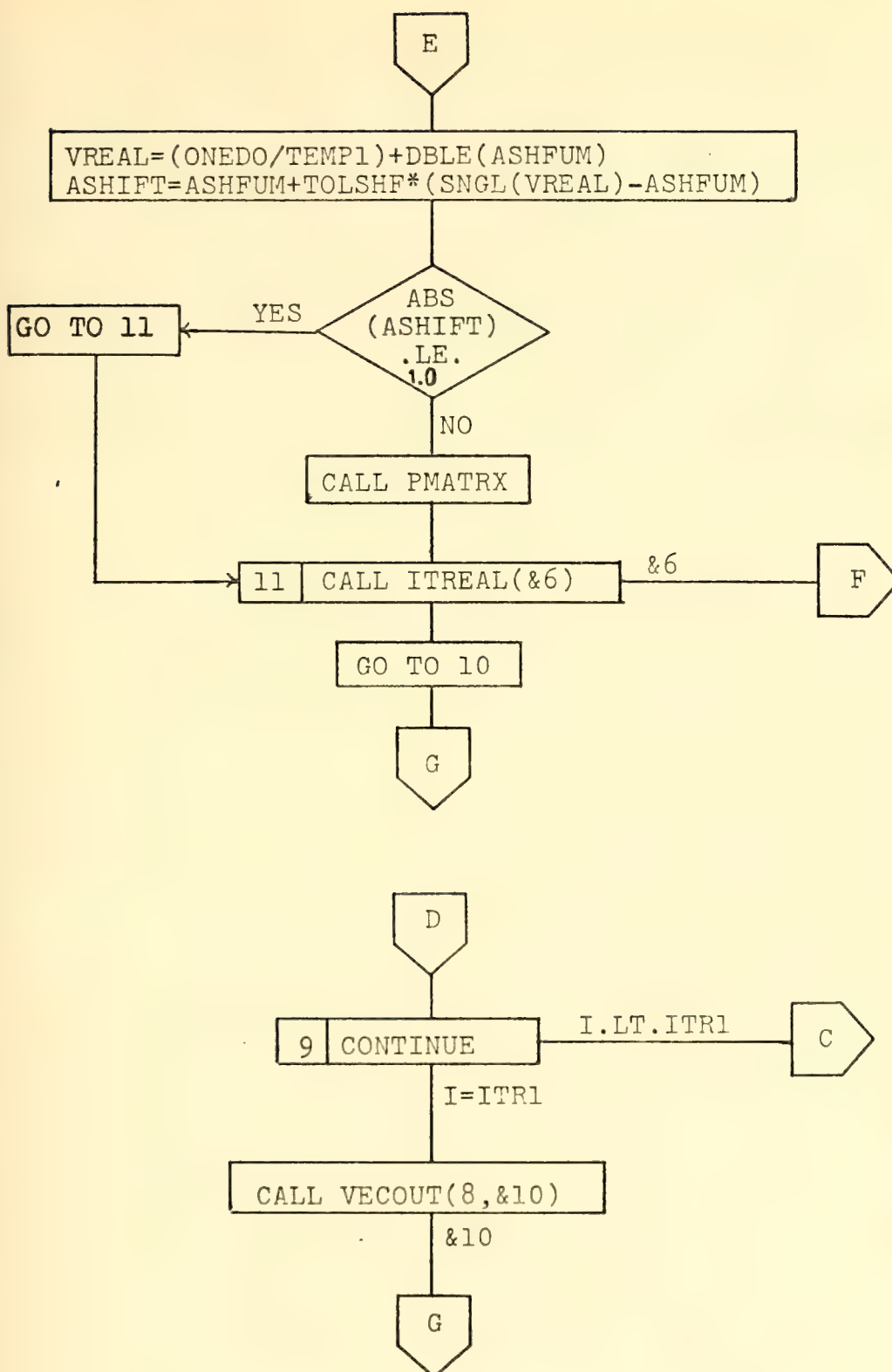


FIGURE 16 continued





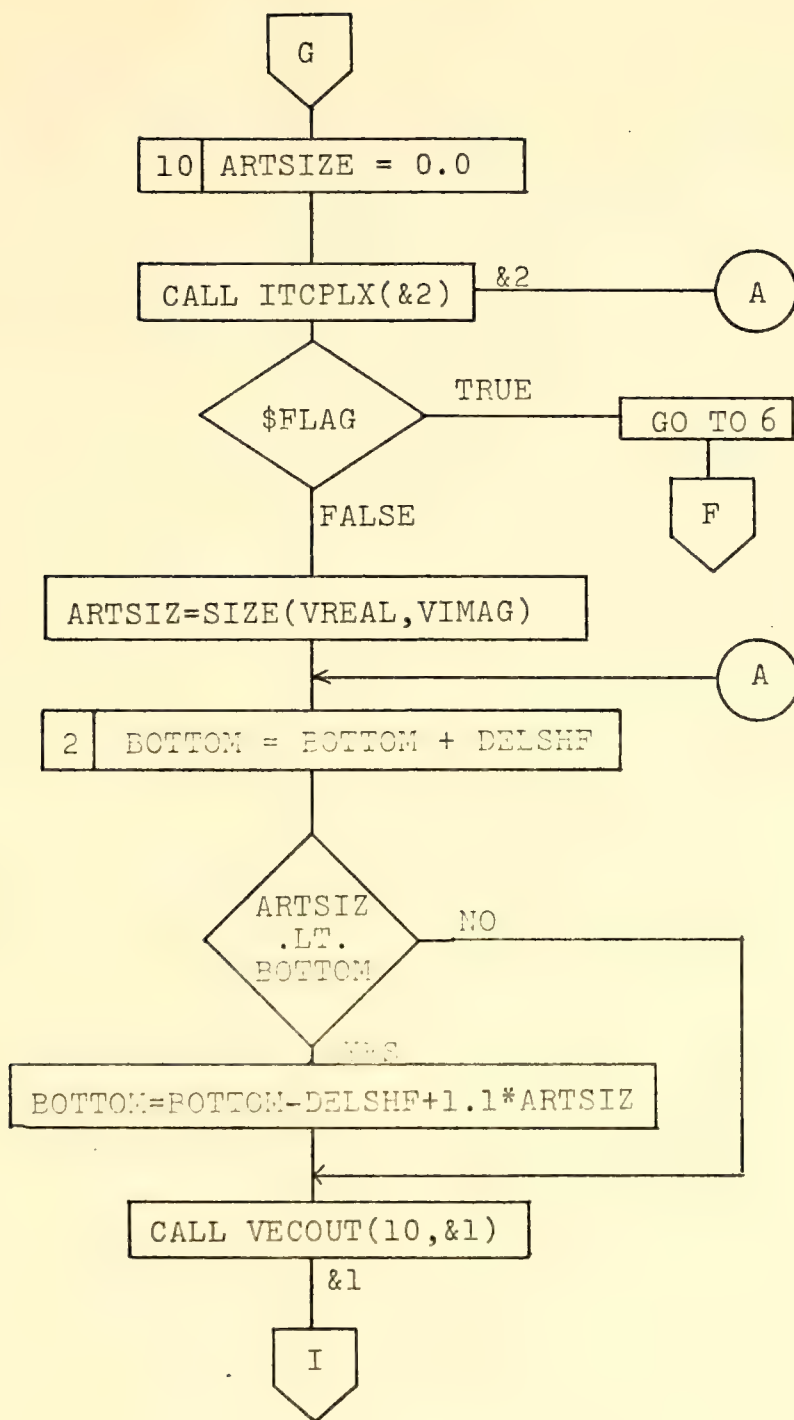


FIGURE 16 continued



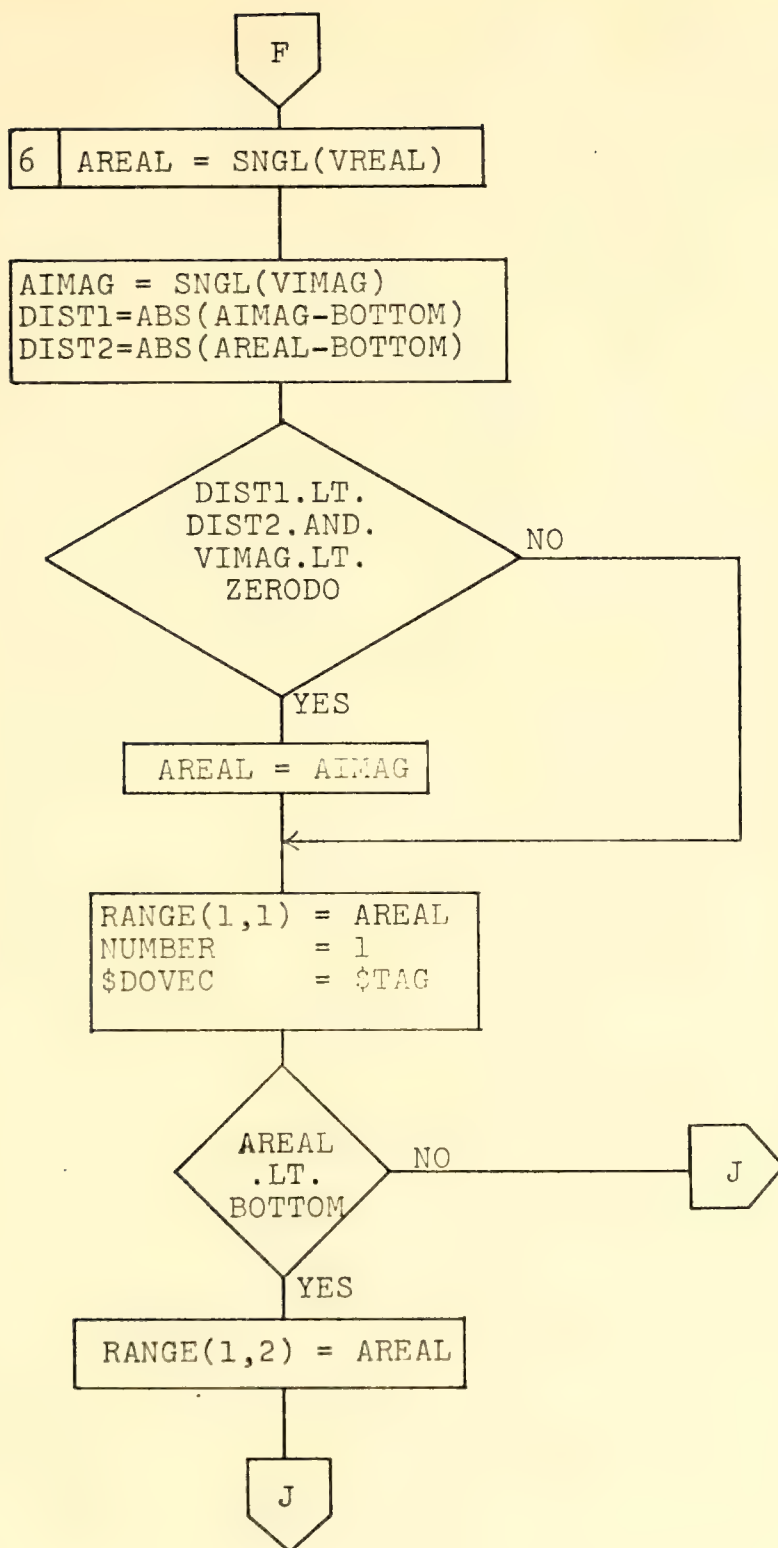


FIGURE 16 continued



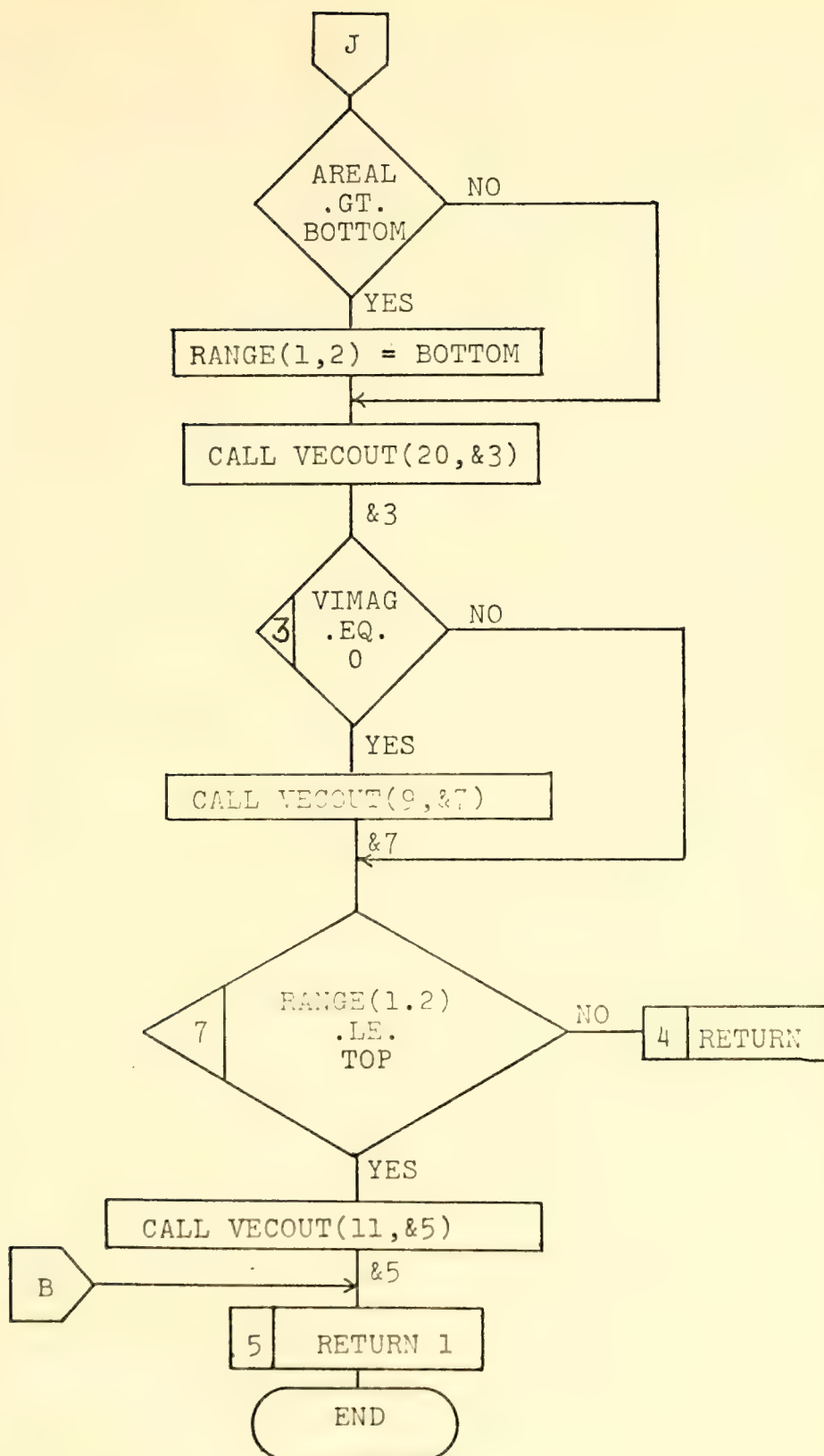


FIGURE 16 continued



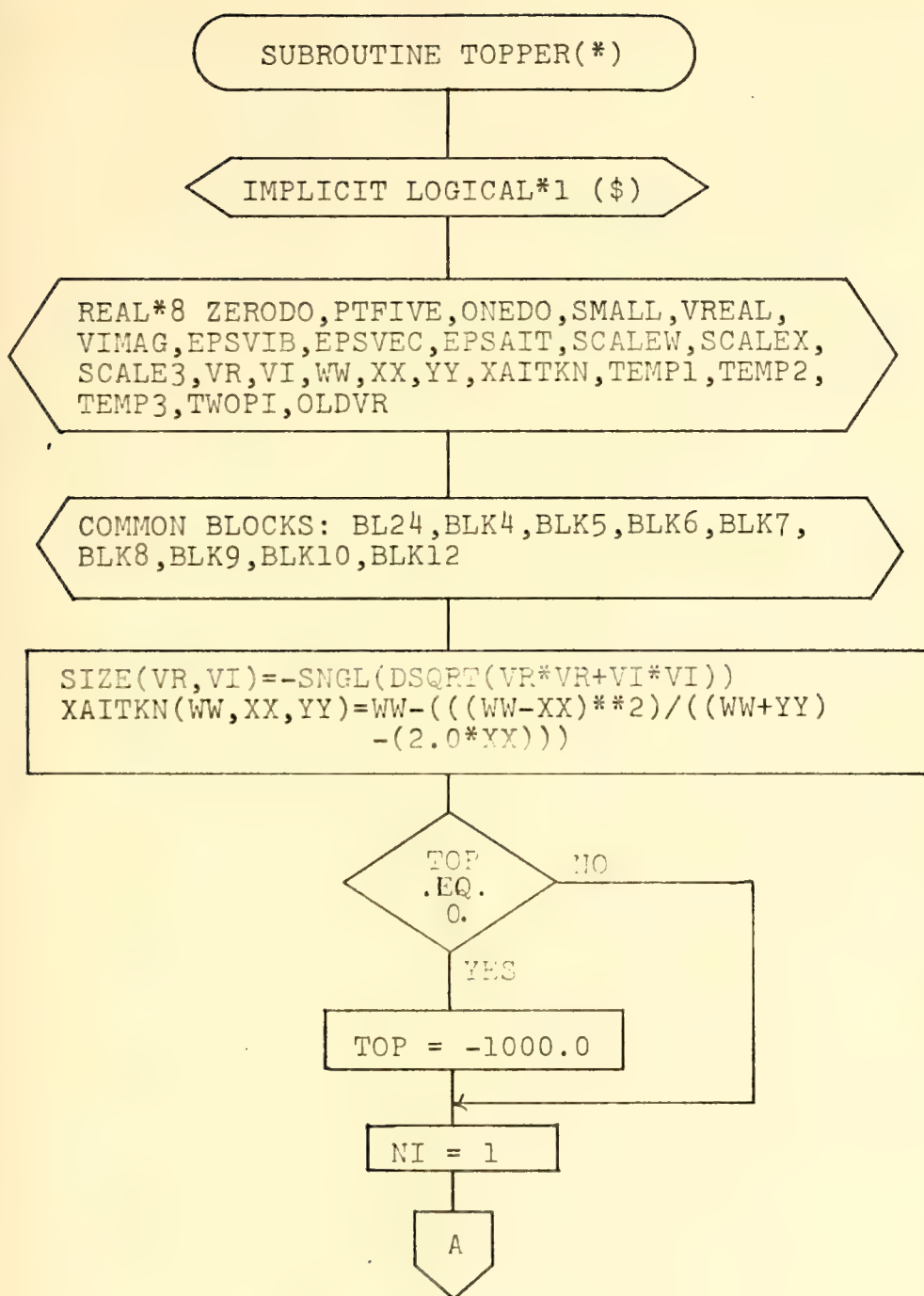


FIGURE 17. SUBROUTINE TOPPER(\*) FLOWCHART





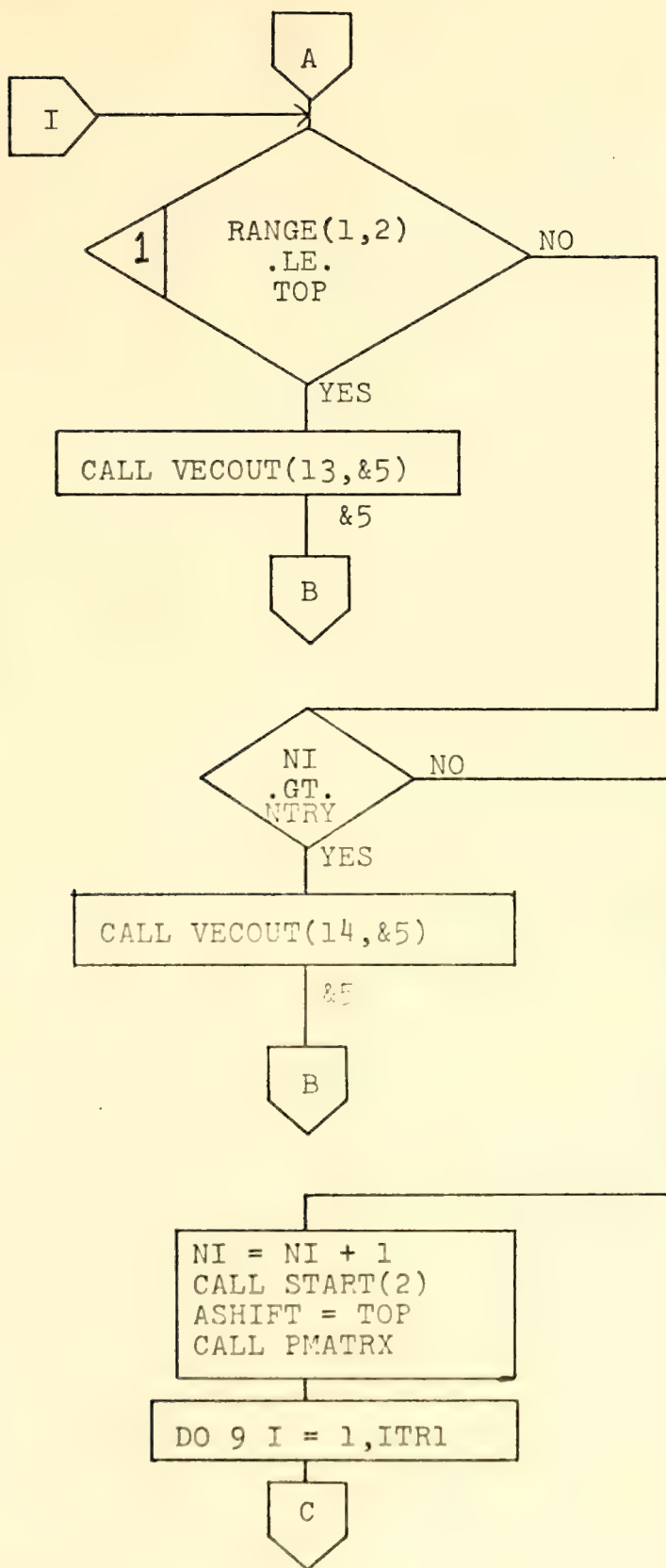


FIGURE 17 continued



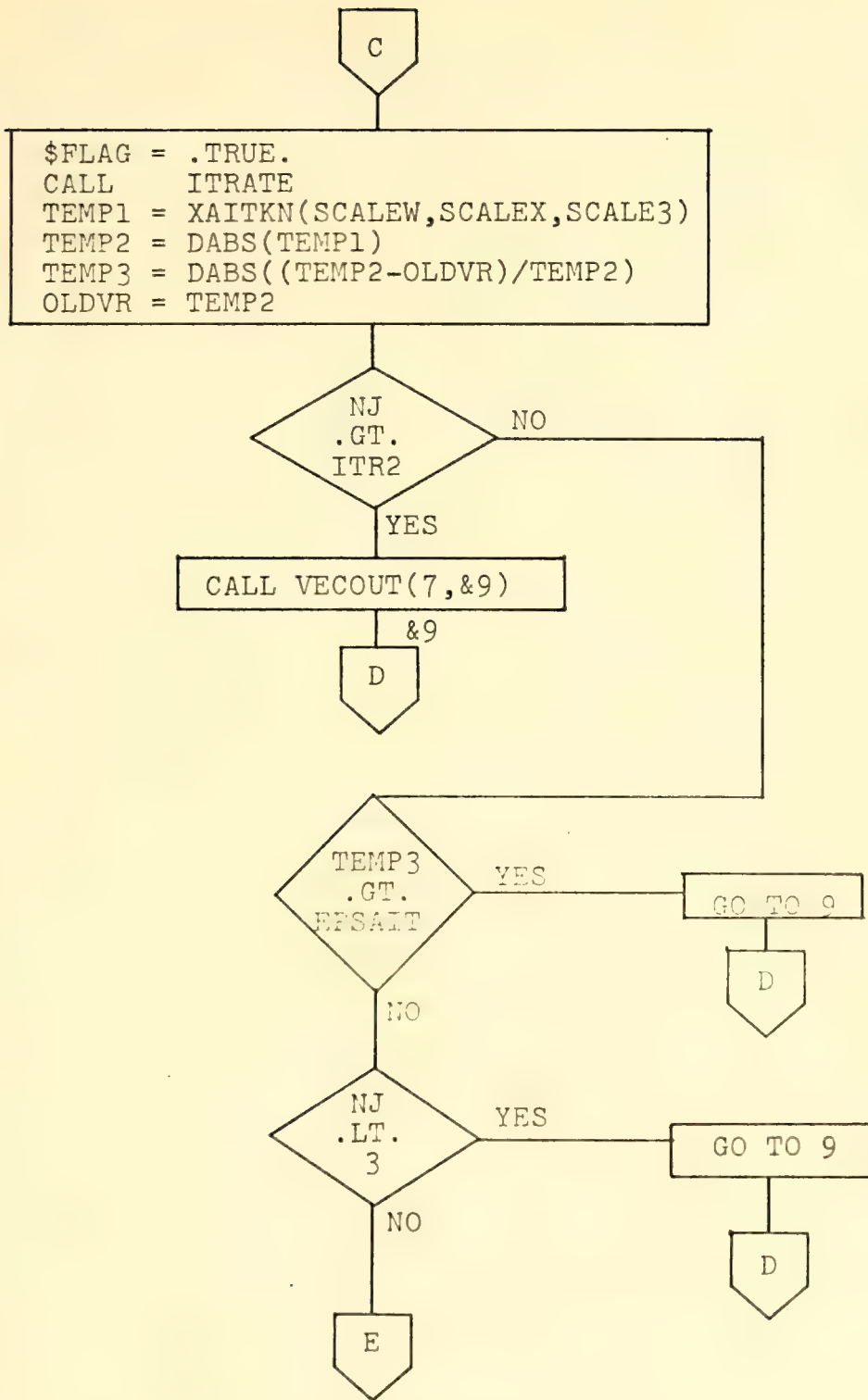


FIGURE 17 continued



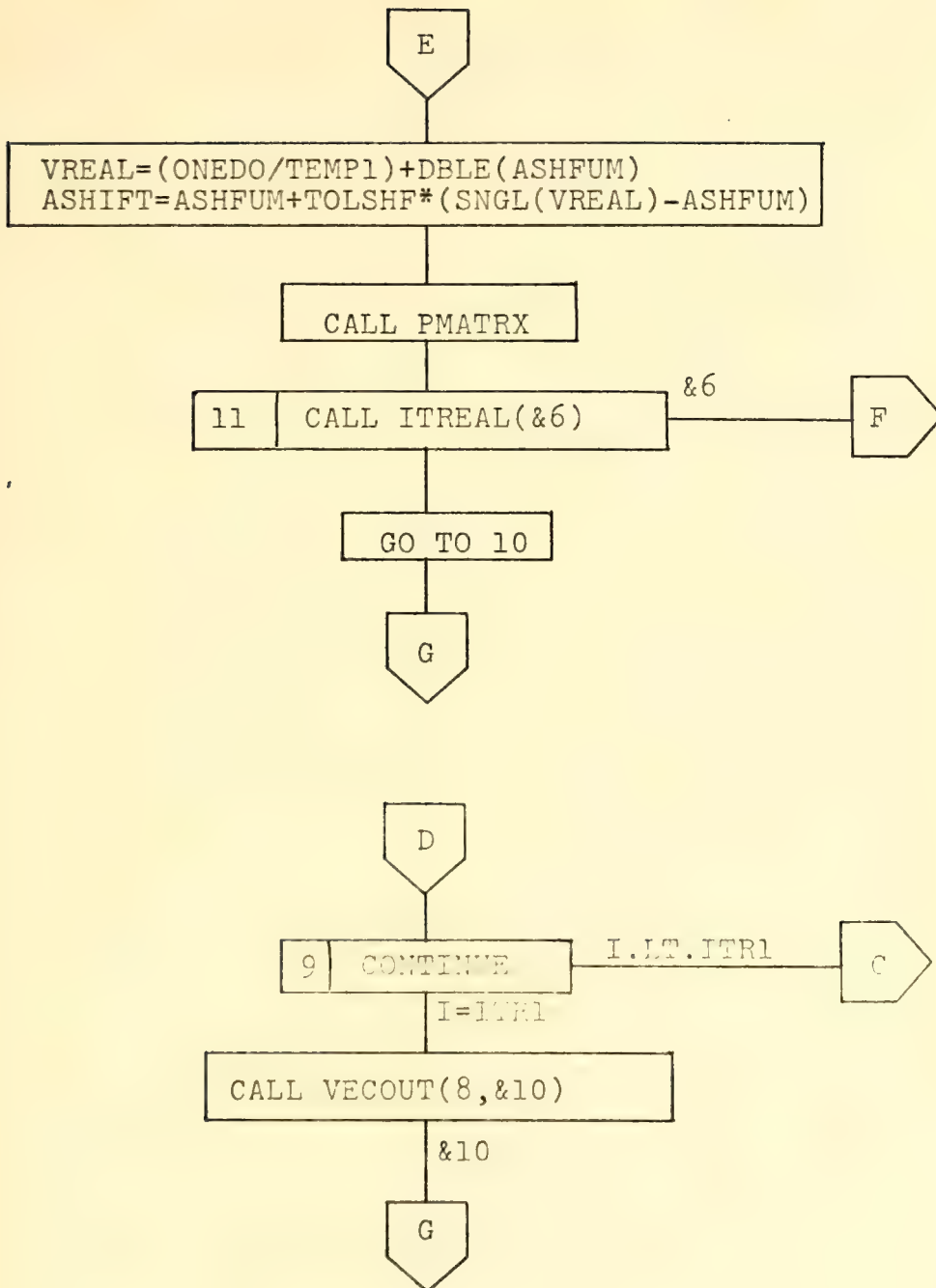


FIGURE 17 continued



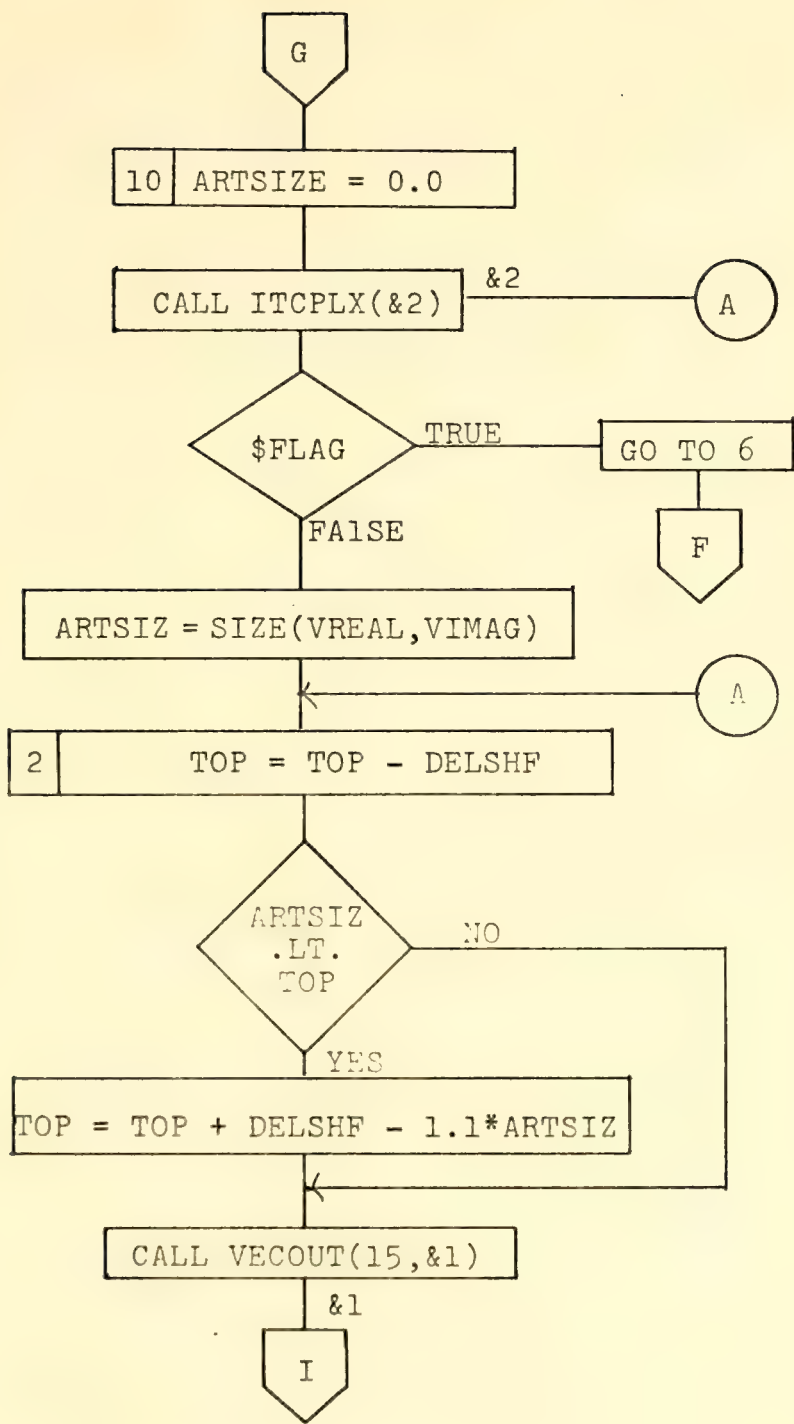


FIGURE 17 continued





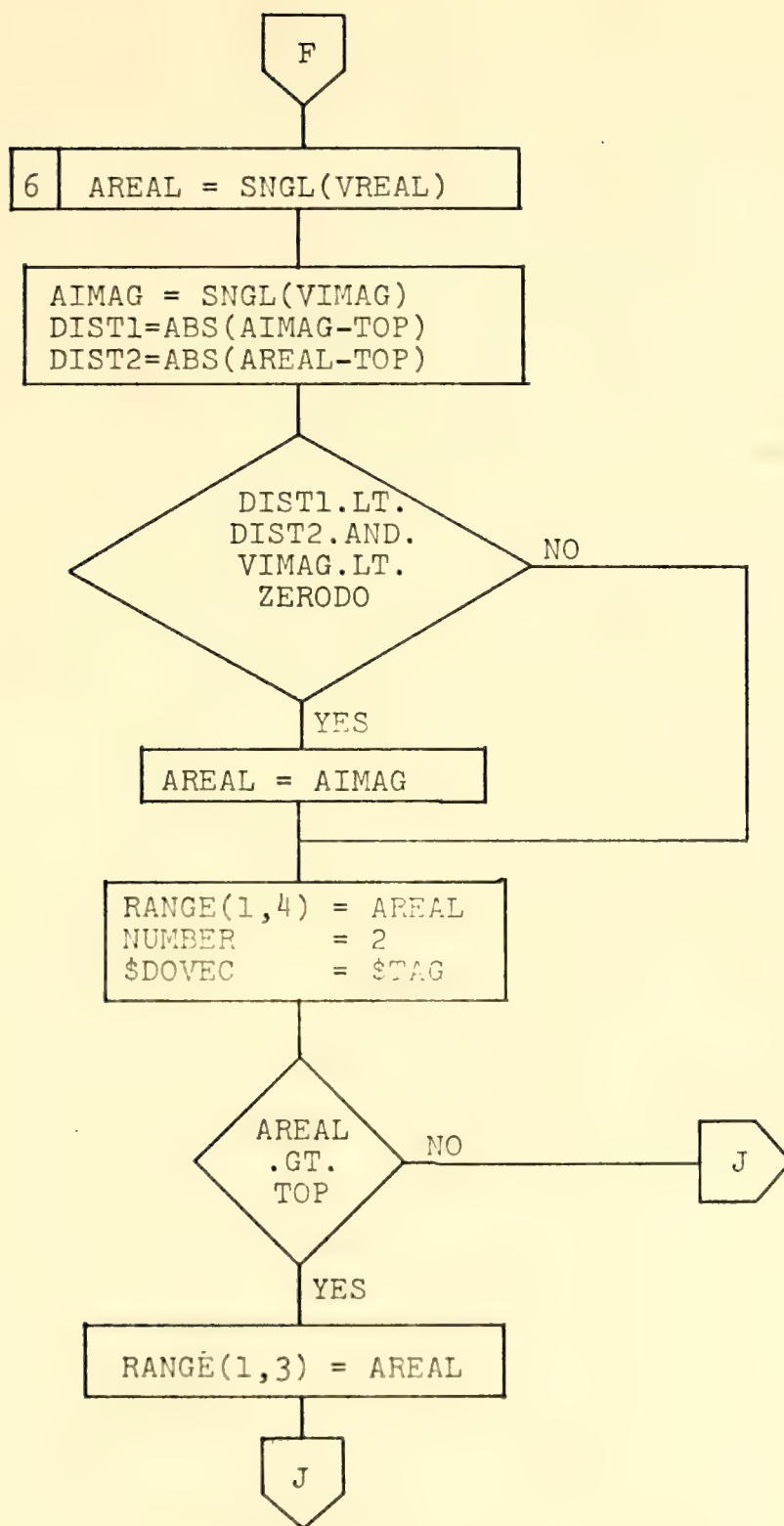


FIGURE 17 continued



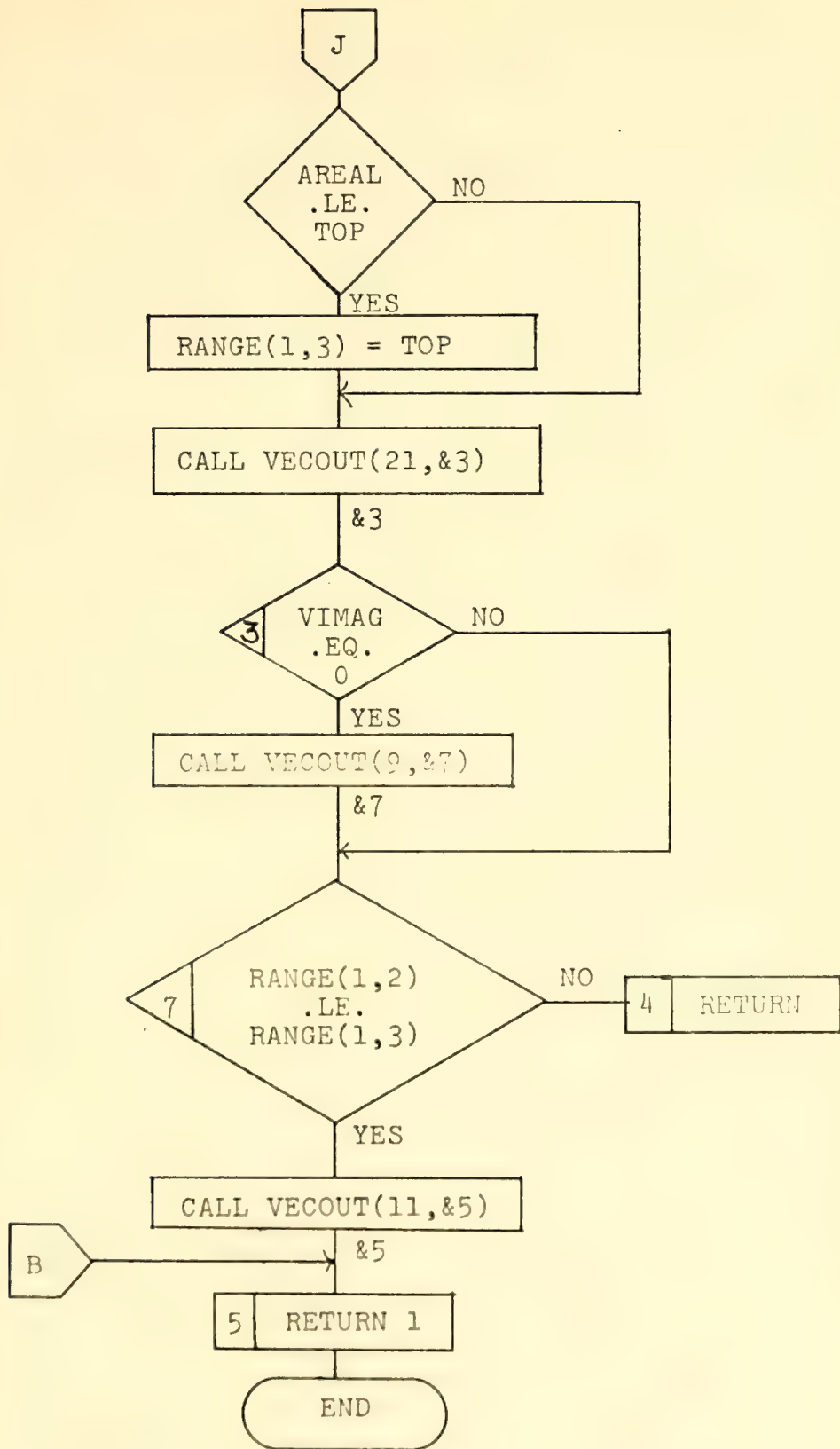


FIGURE 17 continued



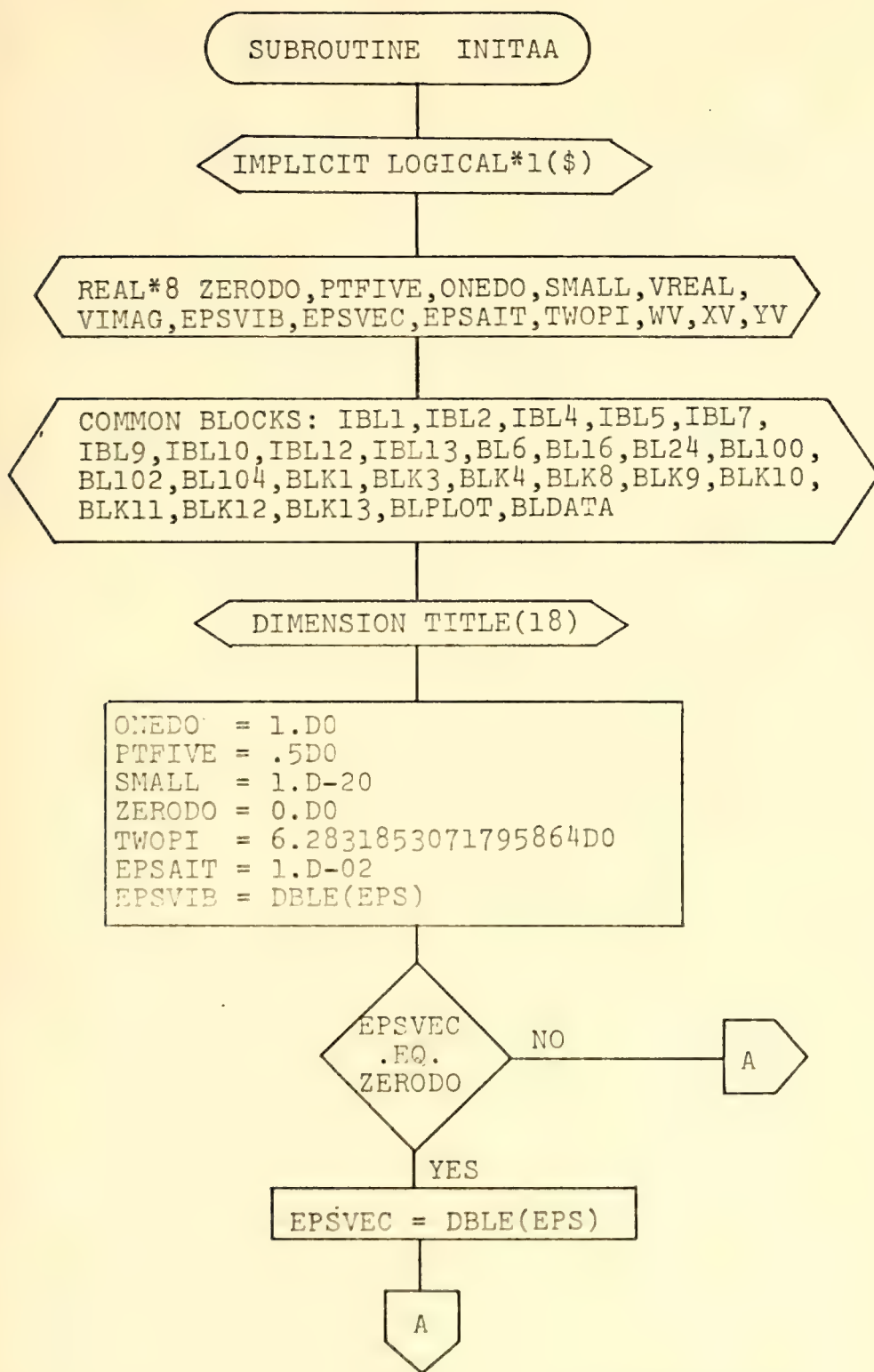


FIGURE 18. SUBROUTINE INITAA FLOWCHART



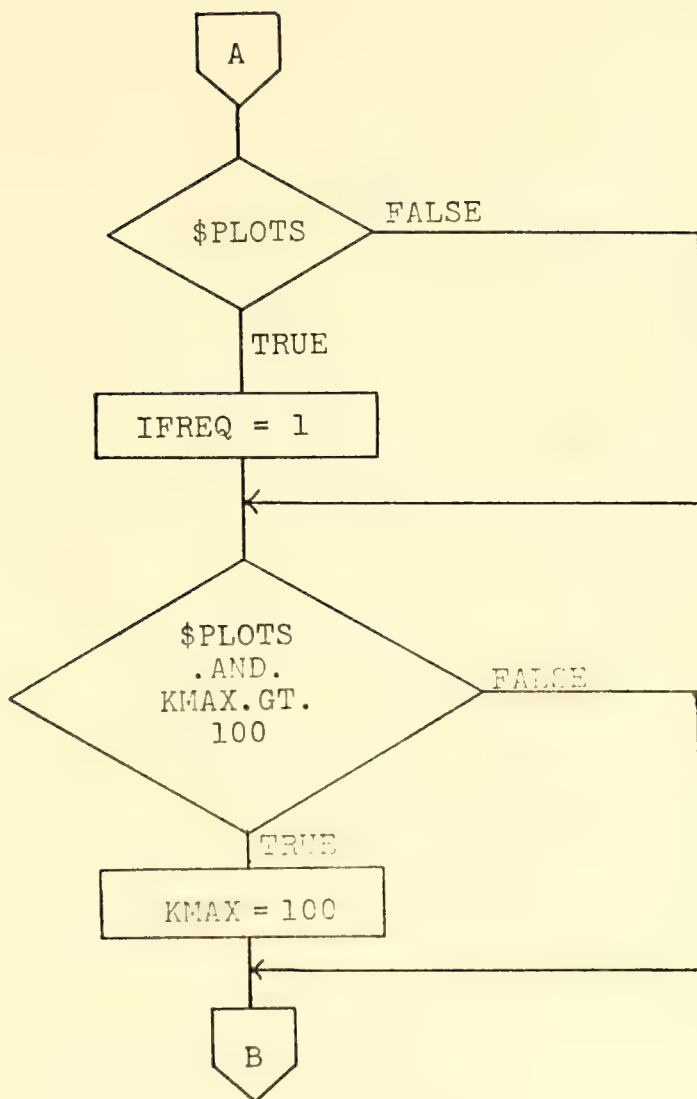


FIGURE 18 continued





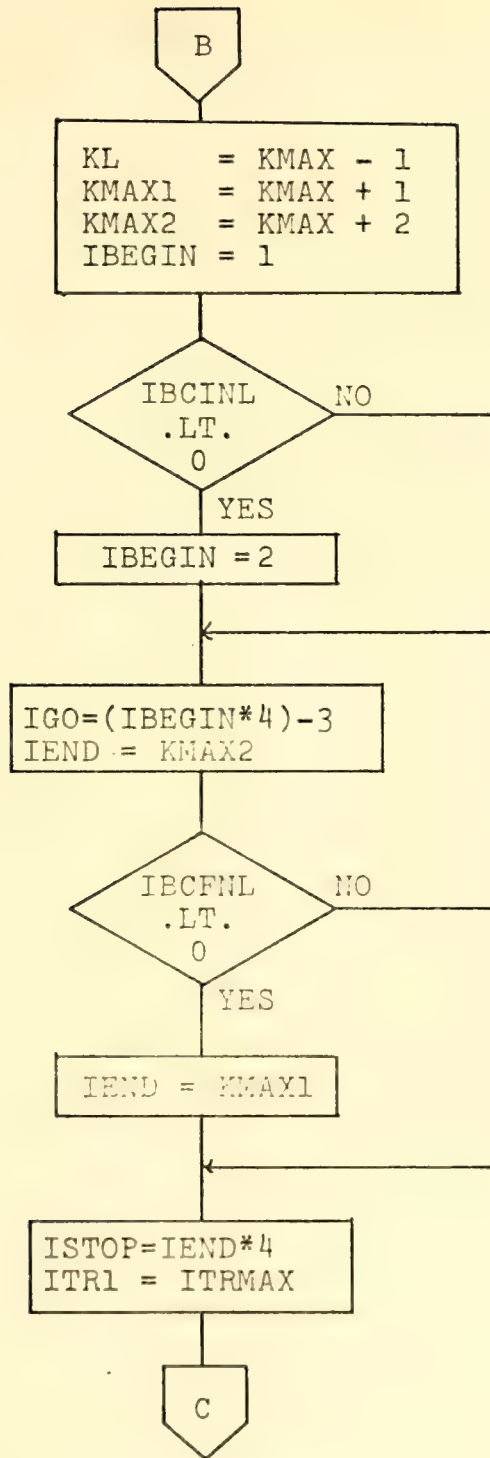


FIGURE 18 continued



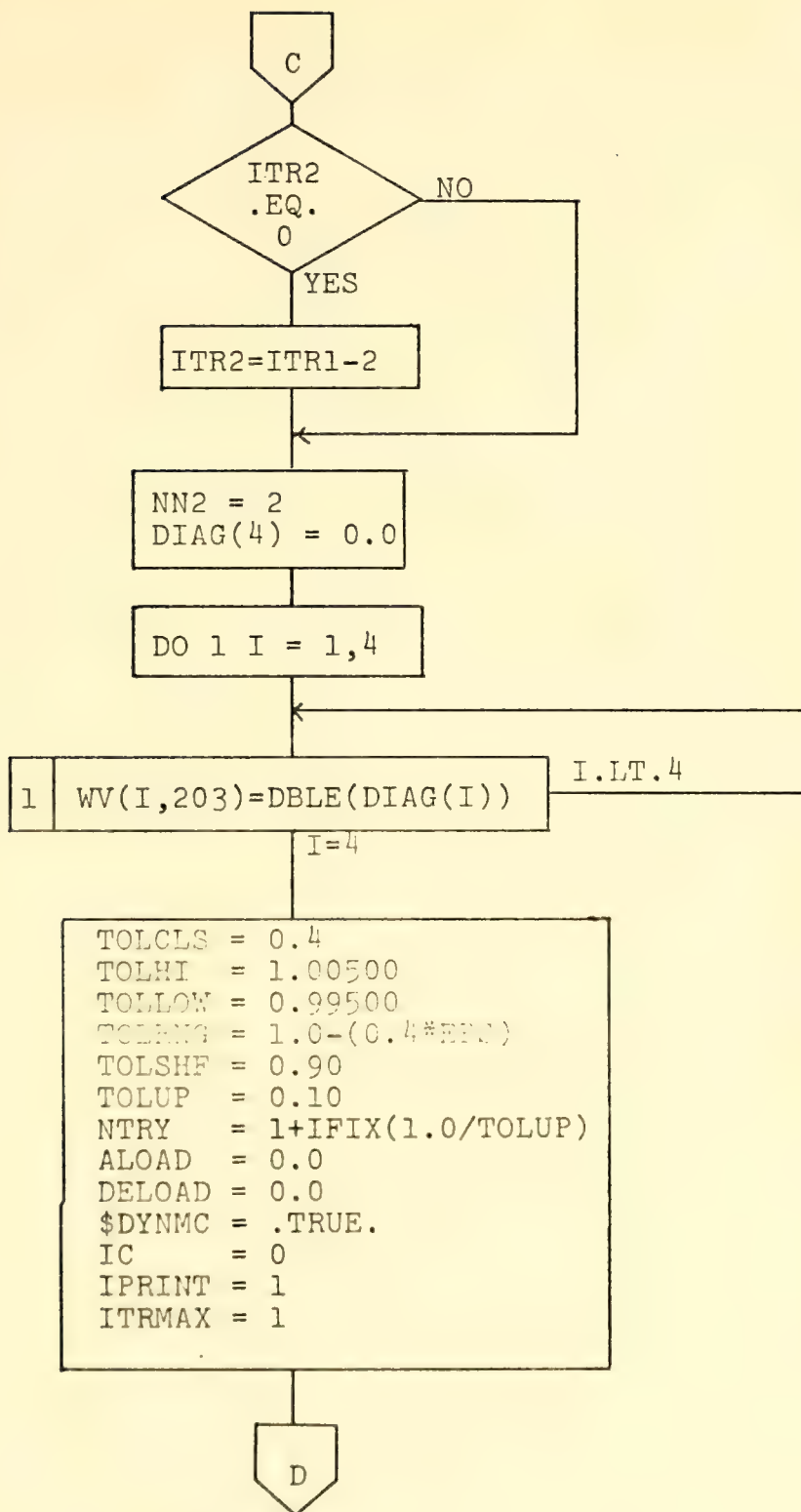


FIGURE 18 continued



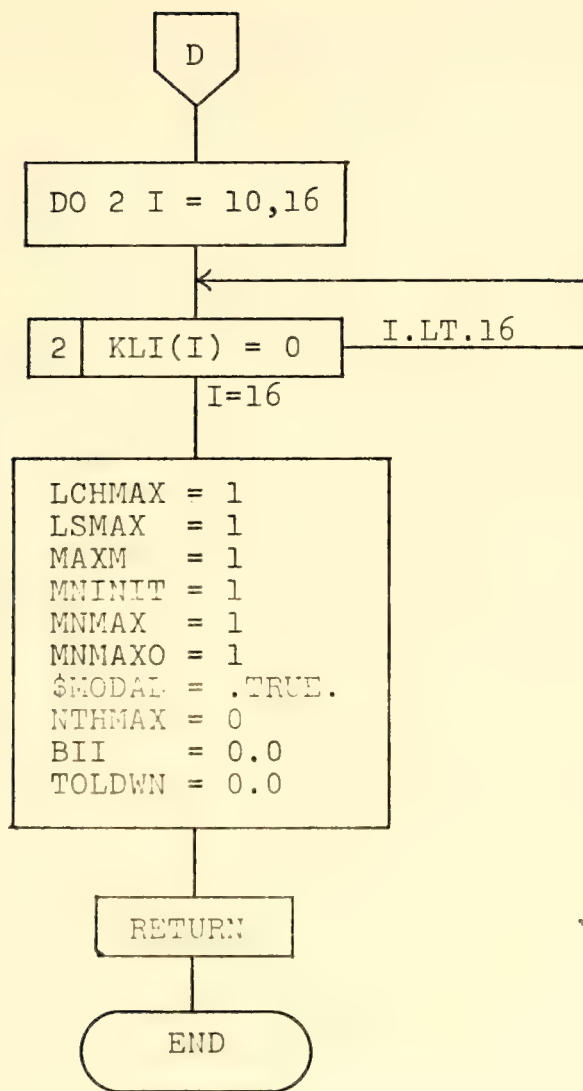


FIGURE 18 continued



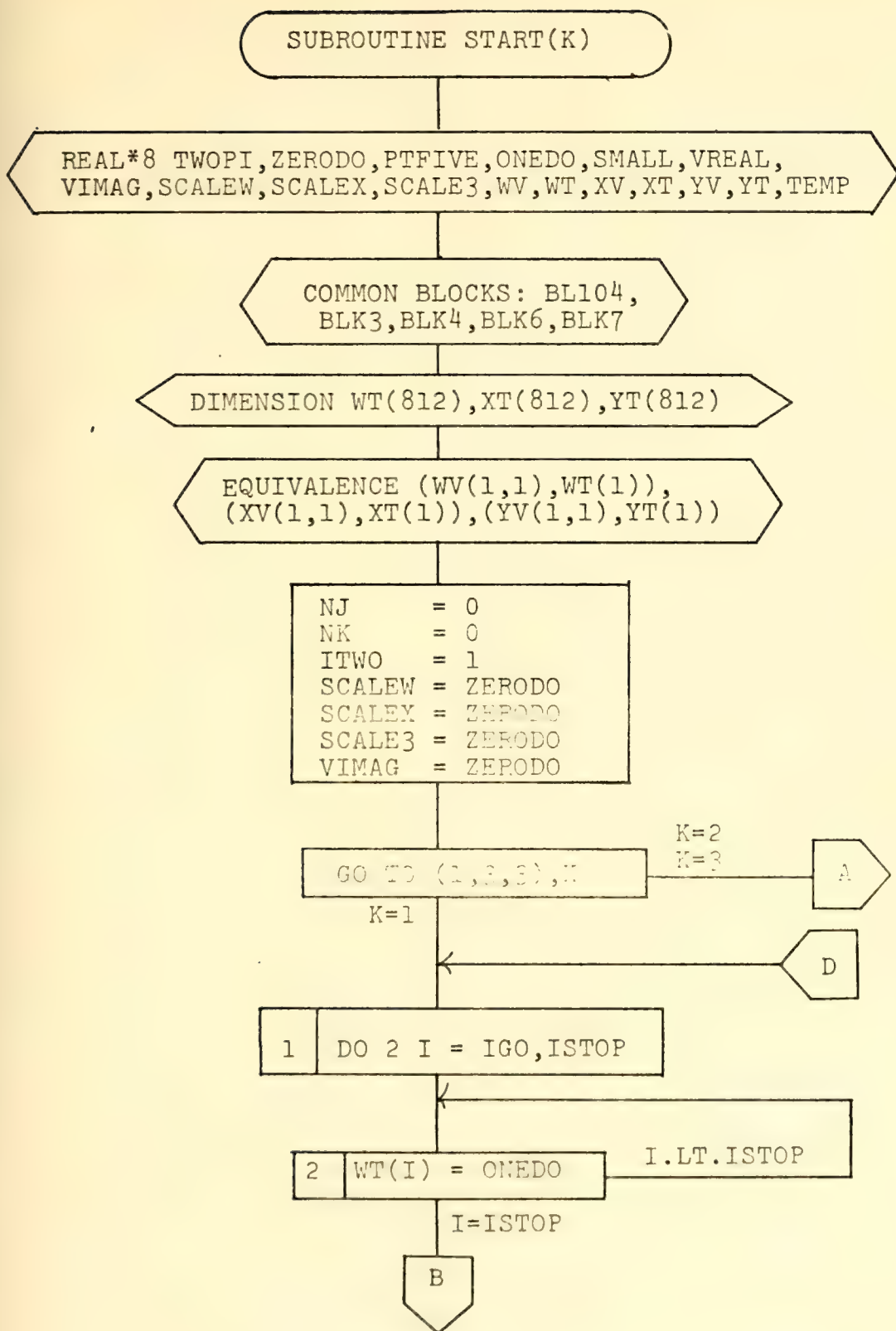


FIGURE 19. SUBROUTINE START(K) FLOWCHART





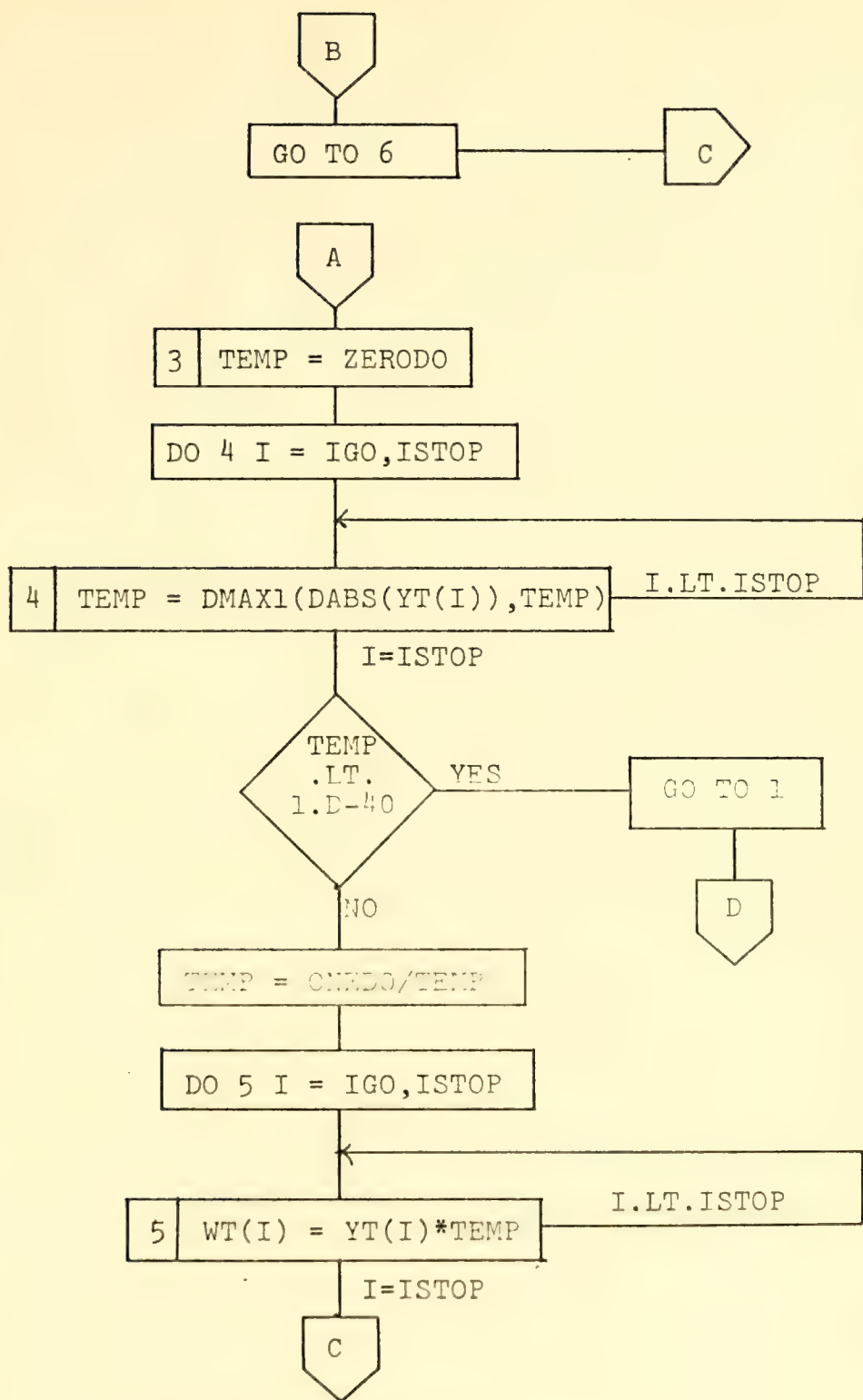


FIGURE 19 continued



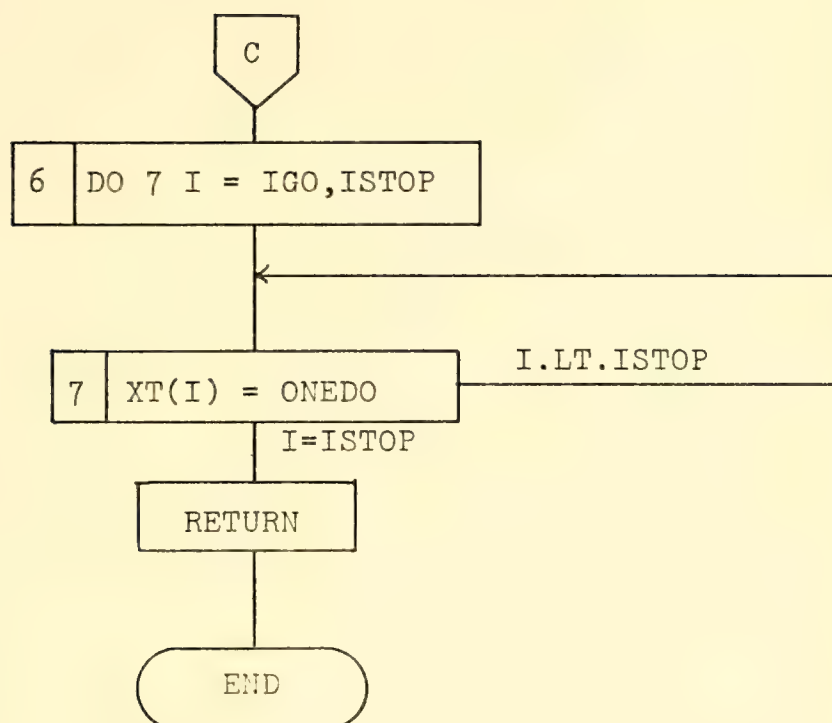


FIGURE 19 continued



```

THE LOWEST FREQUENCY IN HARMONIC 3 IS DETERMINED
DIAGONAL OF MASS MATRIX (MU-BAR) IS ( 1. 1. 1. 0. )
THE CONVERGENCE TEST ON THE VECTOR WILL BE PERFORMED
CONVERGENCE CRITERION IS 0.10D-01

      CONVERGED SOLUTION. ITERATIONS =    18
RAYLEIGH QUOTIENT FREQUENCY =    0.40103772723439D 03 HZ
      ( SOLUTION VECTOR PRINTED HERE )

      3 SHIFT POINT(S) FOR HARMONIC 5 IS(ARE) PROVIDED
DIAGONAL OF MASS MATRIX (MU-BAR) IS ( 0. 0. 1. 0. )
THE CONVERGENCE TEST ON THE VECTOR WILL NOT BE PERFORMED

      CONVERGED SOLUTION. ITERATIONS =     7
RAYLEIGH QUOTIENT FREQUENCY =    0.43982589633472D 02 HZ
      ( OTHER SOLUTIONS FOLLOW )

```

FIGURE 20. Subroutine VECOUT(IND,\*) Print-out Samples



```

THIS IS A FREQUENCY RANGE SEARCH FOR HARMONIC 3

BOTTOM OF RANGE IS 0.7000E 01 HZ      TOP OF RANGE IS 0.1200E 02 HZ
BOTTOM = 0.4900E 02                  TOP = 0.8400E 02

    DIAGONAL OF MASS MATRIX (MU-BAR) IS ( 1. 1. 1. 0. )

    BOTTOM = 0.4900E 02
    LOWEST EIGENVALUE (RANGE(1,1)) = 0.5300E 02
    RANGE(1,2) = 0.5300E 02

    CONVERGED SOLUTION. ITERATIONS = 9

    RAYLEIGH QUOTIENT FREQUENCY = 0.8358963820957D 01 HZ

    ( SOLUTION VECTOR PRINTED HERE )

    TOP = 0.8400E 02
    HIGHEST EIGENVALUE (RANGE(1,4)) = 0.7000E 02
    RANGE(1,3) = 0.7000E 02

    CONVERGED SOLUTION. ITERATIONS = 11

    RAYLEIGH QUOTIENT FREQUENCY = 0.1000000000000E 02 HZ

    ( OTHER SOLUTIONS FOLLOW )

    5 FREQUENCIES HAVE BEEN FOUND

```

FIGURE 20. continued





# Diagnostic Messages:

```

EPS IS VERY SMALL ( 0.10D-09 )

CHECK PERMISSIBLE NUMBER OF ITERATIONS ( 3 )

NO CONVERGENCE TO A REAL EIGENVALUE AFTER 99 ITERATIONS
RAYLEIGH QUOTIENT FREQUENCY = 0.1234567891011D 04 HZ
{THE FREQUENCY IS IMAGINARY}
THE LAST, NEXT TO LAST AND NEXT TO NEXT TO LAST EIGENVALUES ARE
-0.987654321098D 02 -0.987654321005D 02 -0.987656789876D 02

NO CONVERGENCE AFTER 99 ITERATIONS
EIGENVALUE(S) ARE 0.123456D 01 0.123456D-01
THE LAST, NEXT TO LAST AND NEXT TO NEXT TO LAST VECTOR ELEMENT MAXIMUMS ARE
0.123456789102D 03 -0.987654323456D-04 0.123456789345D 04

CONVERGED SOLUTION - TWO REAL EIGENVALUES
AN INCREASE IN THE CONVERGENCE CRITERIA (EPS) AND/OR AN INCREASE IN THE PERMISSIBLE NUMBER OF
ITERATIONS(ITRMAX) IS RECOMMENDED AS A POSSIBLE SOLUTION TO THE PROBLEM OF A POORLY CONDITIONED MATRIX.
ITERATIONS ARE 99
{THE FIRST FREQUENCY IS IMAGINARY}
{BOTH FREQUENCIES ARE IMAGINARY}

CONVERGED SOLUTION - COMPLEX EIGENVALUE
ITERATIONS ARE 99 EIGENVALUE IS 0.123456D 02 0.123456D-02

THE RANGE SIZE HAS BEEN REDUCED
BOTTOM OF RANGE IS NOW 0.123456E 01

UNABLE TO DETERMINE A MINIMUM FREQUENCY AFTER 10 TRIES

THE RANGE SIZE HAS BEEN REDUCED
TOP OF RANGE IS NOW 0.123456E 04

UNABLE TO DETERMINE A MAXIMUM FREQUENCY AFTER 10 TRIES
RANGE HAS BEEN COVERED

```

FIGURE 20. continued



APPENDIX D  
SOME EXPERIENCE AND  
RECOMMENDATIONS ON USING SATANS-III

Selecting the program mode which determines the minimum frequency for a given Fourier harmonic (circumferential wave number),  $KEY1 = 0$ , causes the vector test to be performed. To determine the minimum eigenvalue without performing the vector test, the supplied shift point program mode should be used,  $KEY1 = 1$ , with the shift point set to zero and  $KEY3 = 0$ .

The vector convergence test is intended for use only when there is some difficulty experienced in extracting a particular mode.

Three shift points per Fourier harmonic may be supplied by the user,  $KEY1 = 3$ , on one card. The starting vector for the first value is all ones. The starting vector for the one or two subsequent values is determined by the pseudo sweeping technique developed for subroutine RANGER. However, if one of those values is zero, the starting vector will be all ones. Three shift points enable a difficult mode to extract to be bracketed.

In the frequency range search operation the user supplies the upper and lower bounds of the range. In the lower modes the iteration process is sensitive to the choice of starting vectors. On infrequent occasions temporary convergence to one of two closely spaced eigenvalues



might cause a mode to be missed. This may be avoided by making the convergence criterion very tight, using the vector test and using a large number of iterations. This is a high price to pay for the average case and the supplied shift point mode of the program is a more economical approach and the recommended course of action. The simultaneous determination of the mode shapes provides an easy method of spotting if and which mode is missing.

Because the starting vectors are biased toward the lower frequencies the range bounds supplied by the user may not be the range actually searched by SATANS-III. The range actually searched is from BOTTOM to TOP as subsequently determined by subroutine RANGER. If, for example, a user specifies the range 1000 to 2000 Hz the bias toward lower frequencies might cause the actual search range to be from 800 to 1500 Hz. For this reason it is recommended that at least the upper bound be over-estimated to insure the complete search of the desired range. A good rule of thumb for the lower frequencies is an over-estimation of 50 to 100%. If a range is very small it might be searched more economically by the supplied shift method.

The only restriction on output is that only modal data is available. Since only one harmonic at a time is involved this was not thought to be a severe restriction. No matter what the input value is NTHMAX is subsequently set to zero.

In plotting output it must be remembered that KMAX must be less than 101.



Some difficulty has been encountered in nondimension-  
alizing and it is recommended that it not be employed.  
To avoid this difficulty set SIGO, ELAST, TKN and CHAR  
(input card three) to 1.0. Adjustments in the relative  
magnitude of the shift point values may be made using TEE0.

It was discovered that using the Naval Postgraduate  
School's IBM-360 under OS/MVT, Release 18, and the catalogued  
procedure FORTHCLG overflowed the linkage editor symbol  
table. This was corrected by the specification PARM.LINK  
='SIZE=(150K,20480),LIST,MAP',REGION.LINK=200K





APPENDIX E

SATANS-III INPUT CARD DATA GUIDE

<u>Card</u>	<u>Columns</u>	<u>Format</u>	<u>Item</u>	<u>Example</u>	<u>Meaning</u>
1	1-72	18A4	TITLE	-	Enter any 72 characters
-----					
2	1- 5	I5	NO	1	The problem number, 0<NO<100000.
2	6- 8	L3	\$DYNMC	F T	\$DYNMC = F for a static analysis. \$DYNMC = T for dynamic or vibration analyses.
2	9-10	L2	\$VIBES	F T	\$VIBES = F for dynamic and static analyses. \$VIBES = T for a vibration analysis.
2	11-15	I5	IMODE	0 1	For no modal output data. For modal output data.
2	16-20	I5	NDIMEN	0 1	Dimensional output data. Nondimensional output data.
2	21-25	I5	NTHMAX	8	Summed solution will be printed at NTHMAX meridians, 0<=NTHMAX<=36. Not applicable in a vibration analysis.
2	26-30	I5	IFREQ	3	Solution will be printed at the first and last stations and every IFREQth intermediate station.
2	31-35	I5	IPRINT	2	Every IPRINTth converged solution will be printed. Not applicable in a vibration analysis.



<u>Card</u>	<u>Columns</u>	<u>Format</u>	<u>Item</u>	<u>Example</u>	<u>Meaning</u>
2	36-40	I5	IBCINL	-1 0	If the shell has a pole at the first station. If the shell does not have a pole at the first station.
2	41-45	I5	IBCFNL	-1 0	If the shell has a pole at the last station. If the shell does not have a pole at the last station.
2	46-50	I5	KMAX	25	The number of meridional stations. KMAX<201 If plots are desired KMAX<101.
2	51-55	I5	MNMAX	7	The number of Fourier series coefficients used to describe the initial conditions, pressure and thermal loads. Not applicable in a vibration analysis.
2	56-60	I5	MAXM	8	The number of Fourier series coefficients used in describing the solution. MAXM>=MNMAX, MAXM<11 and MAXM * KMAX <201. Not applicable in a vibration analysis.
2	61-65	I5	LSMAX	1 99 750	For a linear analysis. Use many load steps for a nonlinear static analysis. For a dynamic analysis LSMAX is the number of time increments, where $LSMAX = T_{max} / \Delta T$ . Not applicable in a vibration analysis.
2	66-70	I5	LCHMAX	3	The number of load step size reductions in a static analysis. Recommended range is 2-4.



<u>Card</u>	<u>Columns</u>	<u>Format</u>	<u>Item</u>	<u>Example</u>	<u>Meaning</u>
				0	For a dynamic analysis. Not applicable in a vibration analysis
2	71-75	I5	ITRMAX	1 30	For a linear analysis. Maximum number of iterations at a load or time step or for a frequency. Up to 30 for a nonlinear analysis and 99 for a vibration analysis. Adjust as desired.
2	76-80	I5	IC	0  1	Initial conditions. Set to 0 for a static analysis, or for a dynamic analysis for a shell at rest at $T = 0$ . For a dynamic analysis with initial conditions.
-----					
3	1-12	E12.3	NU	0.3	Poisson's ratio, $\nu$ .
3	13-24	E12.3	SIGO	1000.0  1.0	Reference stress level, $\sigma_0$ If the input data is dimensional.
3	25-36	E12.3	ELAST	.3E8  1.0	Reference modulus of elasticity, $E_0$ . If the input data is dimensional.
3	37-48	E12.3	TKN	.4E-2 1.0	Reference thickness, $h_0$ . If the input data is dimensional.
3	49-60	E12.3	CHAR	8.6  1.0	Characteristic shell dimension, $a$ . If the input data is dimensional.



<u>Card</u>	<u>Columns</u>	<u>Format</u>	<u>Item</u>	<u>Example</u>	<u>Meaning</u>
3	61-72	E12.3	TEEO	0.0 .966E-5  10.0	For a static analysis. Reference time, $T_0$ for a dynamic analysis. Nondimensionalizing time in a vibration analysis. Select in accordance with Eq (2).
-----					
4	1-12	E12.3	DELOAD	0.2      .8E-7	For a static analysis DELOAD is the load incre- ment. It remains un- changed until the solu- tion fails to converge in ITRMAX iterations. It is then reduced by a factor of five. A max- imum of LCHMAX reductions will occur before the program terminates. For a dynamic analysis DELOAD is the nondimen- sional time increment. LSMAX such time incre- ments are made. Not applicable in a vibration analysis.
4	13-24	E12.3	EPS	.01	Convergence criterion. Recommended range is .01 to .0001 for most applications. For a vibration analysis a somewhat tighter cri- terion may be desired.

-----

Include as many cards 5 as necessary to specify the NTHMAX  
meridians. If NTHMAX = 0 then card 5 is OMITTED.

5	1-72	6E12.3		10.0	A list of circumferential coordinates, $\theta$ , in degrees and tenths, at which the solution will be printed. There must be at least NTHMAX entries. Not applicable in a vibration analysis.
---	------	--------	--	------	---

-----





Cards 6 through 23 describe the boundary conditions at the first and last stations. The boundary conditions exist on the total variables, not on the individual harmonics. If IBCINL = -1 (an initial pole) then OMIT cards 6 through 14, inclusive. If IBCFNL = -1 (a final pole) then OMIT cards 15 through 23, inclusive. Loadings specified through the edge boundary conditions are taken in the zeroth harmonic only. The column matrix { $\ell$ } is set to zero for all other harmonics. The boundary conditions are dimensional. The format of cards 6 through 23 is 4E16.8.

<u>Card 6,15</u>	<u>Card 7,16</u>	<u>Card 8,17</u>	<u>Card 9,18</u>	
$\begin{bmatrix} \Omega(1,1) \\ \Omega(2,1) \\ \Omega(3,1) \\ \Omega(4,1) \end{bmatrix}$	$\begin{bmatrix} \Omega(1,2) \\ \Omega(2,2) \\ \Omega(3,2) \\ \Omega(4,2) \end{bmatrix}$	$\begin{bmatrix} \Omega(1,3) \\ \Omega(2,3) \\ \Omega(3,3) \\ \Omega(4,3) \end{bmatrix}$	$\begin{bmatrix} \Omega(1,4) \\ \Omega(2,4) \\ \Omega(3,4) \\ \Omega(4,4) \end{bmatrix}$	$\left\{ \begin{matrix} N_s \\ \hat{N}_s \\ \hat{N}_s \theta \\ \hat{Q}_s \\ \phi_s \end{matrix} \right\} +$
<u>Card 10,19</u>	<u>Card 11,20</u>	<u>Card 12,21</u>	<u>Card 13,22</u>	<u>Card 14,23</u>
$\begin{bmatrix} \Lambda(1,1) \\ \Lambda(2,1) \\ \Lambda(3,1) \\ \Lambda(4,1) \end{bmatrix}$	$\begin{bmatrix} \Lambda(1,2) \\ \Lambda(2,2) \\ \Lambda(3,2) \\ \Lambda(4,2) \end{bmatrix}$	$\begin{bmatrix} \Lambda(1,3) \\ \Lambda(2,3) \\ \Lambda(3,3) \\ \Lambda(4,3) \end{bmatrix}$	$\begin{bmatrix} \Lambda(1,4) \\ \Lambda(2,4) \\ \Lambda(3,4) \\ \Lambda(4,4) \end{bmatrix}$	$\left\{ \begin{matrix} U \\ V \\ W \\ M_s \end{matrix} \right\} = \left\{ \begin{matrix} \ell(1) \\ \ell(2) \\ \ell(3) \\ \ell(4) \end{matrix} \right\}$



<u>Card</u>	<u>Column</u>	<u>Format</u>	<u>Item</u>	<u>Example</u>	<u>Meaning</u>
24	1- 2	L2	\$PLOTS	F	Indicates plots are NOT desired.
				T	Indicates plots are desired. KMAX<101
24	3- 4	L2	\$MODAL	F	Indicates plots are for summed solutions only
				T	Indicates plots are for modal solutions only.

For the remainder of card 24 entries, 0 indicates that no plots are desired for the particular item, and 1 indicates that they are desired. All graphs are plotted as the indicated item versus the station number. If a complete plot is desired, insure that IFREQ = 1.

24	5- 6	I2	IRADII	1	Plot the radii as computed by subroutine GEOM.
24	7- 8	I2	IGAMMA	1	Plot $\rho'/\rho$ as computed by subroutine GEOM.
24	9-10	I2	IOMEGS	1	Plot $\omega_s$ as computed by subroutine GEOM.
24	11-12	I2	IOMEGT	1	Plot $\omega_\theta$ as computed by subroutine GEOM.
24	13-14	I2	IDEOMS	1	Plot $\omega'$ as computed by subroutine GEOM.
24	15-16	I2	IBSTIF	1	Plot the stiffness d as computed by subroutine BDB.
24	17-18	I2	IDSTIF	1	Plot the stiffness d as computed by the subroutine BDB.
24	19-20	I2	IBBSTF	1	Plot the stiffness $db/d\zeta$ as computed by subroutine BDB.
24	21-22	I2	IDDSTF	1	Plot the stiffness $dd/d\zeta$ as computed by subroutine BDB.
24	23-24	I2	IPR	1	Plot the normal component of the pressure load.



<u>Card</u>	<u>Column</u>	<u>Format</u>	<u>Item</u>	<u>Example</u>	<u>Meaning</u>
24	25-26	I2	IPS	1	Plot the meridional component of the pressure load.
24	27-28	I2	IPT	1	Plot the circumferential component of the pressure load.
24	29-30	I2	ITT	1	Plot the thermal load.
24	31-32	I2	IMT	1	Plot the thermal moment.
24	33-34	I2	IDTT	1	Plot $d/d\xi$ of the thermal load.
24	35-36	I2	IDMT	1	Plot $d/d\xi$ of the thermal moment.
24	37-38	I2	INS	1	Plot the meridional membrane force distribution.
24	39-40	I2	INTH	1	Plot the circumferential membrane force distribution
24	41-42	I2	INSTH	1	Plot the meridio-circumferential membrane force distribution.
24	43-44	I2	IQS	1	Plot the transverse force distribution.
24	45-46	I2	IMS	1	Plot the meridional moment distribution.
24	47-48	I2	IMTH	1	Plot the circumferential moment distribution.
24	49-50	I2	IMSTH	1	Plot the meridio-circumferential moment distribution.
24	51-52	I2	IU	1	Plot the meridional displacement distribution.
24	53-54	I2	IV	1	Plot the circumferential displacement distribution.



<u>Card</u>	<u>Column</u>	<u>Format</u>	<u>Item</u>	<u>Example</u>	<u>Meaning</u>
24	55-56	I2	IW	1	Plot the normal displacement distribution.
24	57-58	I2	IPHIS	1	Plot the meridional rotation distribution.
24	59-60	I2	IPHIT	1	Plot the circumferential rotation distribution.
24	61-62	I2	IPHI	1	Plot the meridio-circumferential rotation distribution.

-----

Cards 24A and 24B are only included in a SATANS-III vibration analysis and then only when \$VIBES is TRUE. If \$VIBES is FALSE (no vibration analysis to be performed) then cards 24A and 24B are OMITTED.

24A	1- 5	I5	IVIBES	4	The number of vibration analyses = the number of cards 24B to be included.
24A	6- 9	4F1.0	DIAG	1110	The elements of the diagonal of the mass matrix $\mu$ -bar. The 4th element is subsequently automatically set to 0.
24A	10-25	F16.8	EPSVEC	1.E-4	Vector test convergence criterion. If EPSVEC is left = 0.0 it is subsequently automatically set to EPS (card 4).
24A	26-30	I5	ITR2	50	ITR2 is a troubleshooting aid. If ITR2 is entered every ITR2 iteration the Rayleigh quotient eigenvalue will be printed. If ITR2 is left = 0 it is subsequently automatically set to print the next to last solution in the event of non-convergence.

-----





<u>Card</u>	<u>Column</u>	<u>Format</u>	<u>Item</u>	<u>Example</u>	<u>Meaning</u>
-------------	---------------	---------------	-------------	----------------	----------------

Include one card 24B for each vibration analysis desired.

The analyses are done one harmonic at a time and all input information is stated on one card per harmonic. The total number of cards 24B = IVIBES. The information is stored in two arrays of three elements each, MODVIB(J,K) and SHFTPT(J,K) where J=1(1)3 for the Kth analysis and K=1(1)IVIBES.

24B	1- 5	I5	KEY1	0	The lowest frequency in the circumferential Fourier harmonic mode specified by KEY2 will be determined. The vector convergence test will automatically be performed.
				1,2, or 3	1,2 or 3 shift points are supplied by the user. Vector convergence test not performed unless so specified.
				4	A frequency range search will be performed for the mode specified by KEY2.
24B	6-10	I5	KEY2	8	The Fourier harmonic in the circumferential direction for which the vibration analysis will be performed.
24B	11-15	I5	KEY3	0	The vector convergence test will not be performed if KEY1=1,2,3 or 4.
				1	The vector convergence test will be performed.
24B	16-30	F15.8		1.E2	If KEY1=1,2 or 3 this is the first shift point. If KEY1=4 this is the lower bound of the frequency range to be searched. Units are Hz (CPS).



<u>Card</u>	<u>Column</u>	<u>Format</u>	<u>Item</u>	<u>Example</u>	<u>Meaning</u>
24B	31-45	F15.8		2.E4	If KEY1=2 or 3 this is the second shift point. If KEY1=4 this is the upper bound of the frequency range search. Units are Hz (CPS).
24B	46-60	F15.8		4.E3	Applicable only if KEY1=3 in which case this is the third shift point. Units are Hz(CPS).
-----					
25	1- 2	I2	IRNAGN	0	Indicates this is the only run by the program.
				1	Indicates another run by the program is to be made. Another set of input data cards must be supplied. The user supplied sub-routines remain the same. The new set of input cards is placed after this card.



## APPENDIX F

### INPUT AND OUTPUT

#### PART ONE

#### USER SUPPLIED PROGRAMS

To use SATANS-III the user must supply 6 programs. The first is a simple 6 card main program that passes control to SATANS.

The second, subroutine GEOM, computes the nondimensional geometry functions of the shell. The correspondence between the nondimensional geometry functions and FORTRAN variables is as follows:

$$\text{DEL} = \Delta = S_T / (a(KMAX-1))$$

$$R(K) = \rho_K = (r/a)_K$$

$$\text{GAM}(K) = \gamma_K = (\rho'/\rho)_K = ((dr/ds)/(r/a))_K$$

$$\text{OMT}(K) = (\omega_\theta)_K = (a/R_\theta)_K$$

$$\text{OMXI}(K) = (\omega_s)_K = (a/R_s)_K$$

$$\text{DEOMX}(K) = (\omega'_s)_K = (a^2 d(R_s^{-1})/ds)_K$$

$$\text{MASS}(K) = \mu_K$$

for K = 1(1)KMAX



The third, subroutine BDB, computes the nondimensional stiffness quantities  $b$ ,  $b'$ ,  $d$  and  $d'$ . Correspondence between the stiffness quantities and FORTRAN variables is as follows:

$$B = b_K = \int E_y d\zeta / ((1-\nu^2)E_o h_o)$$

$$DB = b'_K = a(dB/ds)_K$$

$$D = d_K = \int \zeta^2 E_y d\zeta / ((1-\nu^2)E_o h_o^3)$$

$$DD = d'_K = a(dD/ds)_K$$

The last three subroutines, PLOAD, TLOAD and INITL compute the nondimensional loads applied to the shell, thermal loads applied to the shell and, in the case of a dynamic analysis, the initial conditions on  $z$  and  $\frac{\partial z}{\partial t}$ . Details on these subroutines are given in Refs. 1 and 2.

APPENDIX G contains example input routines for a vibration analysis.





SAMPLE OUTPUT

--PROBLEM NUMBER 7--

CIRCULAR CYLINDRICAL SHELL WITH A LINEARLY VARYING THICKNESS

--INPUT DATA RECORD--

THE BOUNDARY CONDITIONS ARE:

AT THE INITIAL EDGE

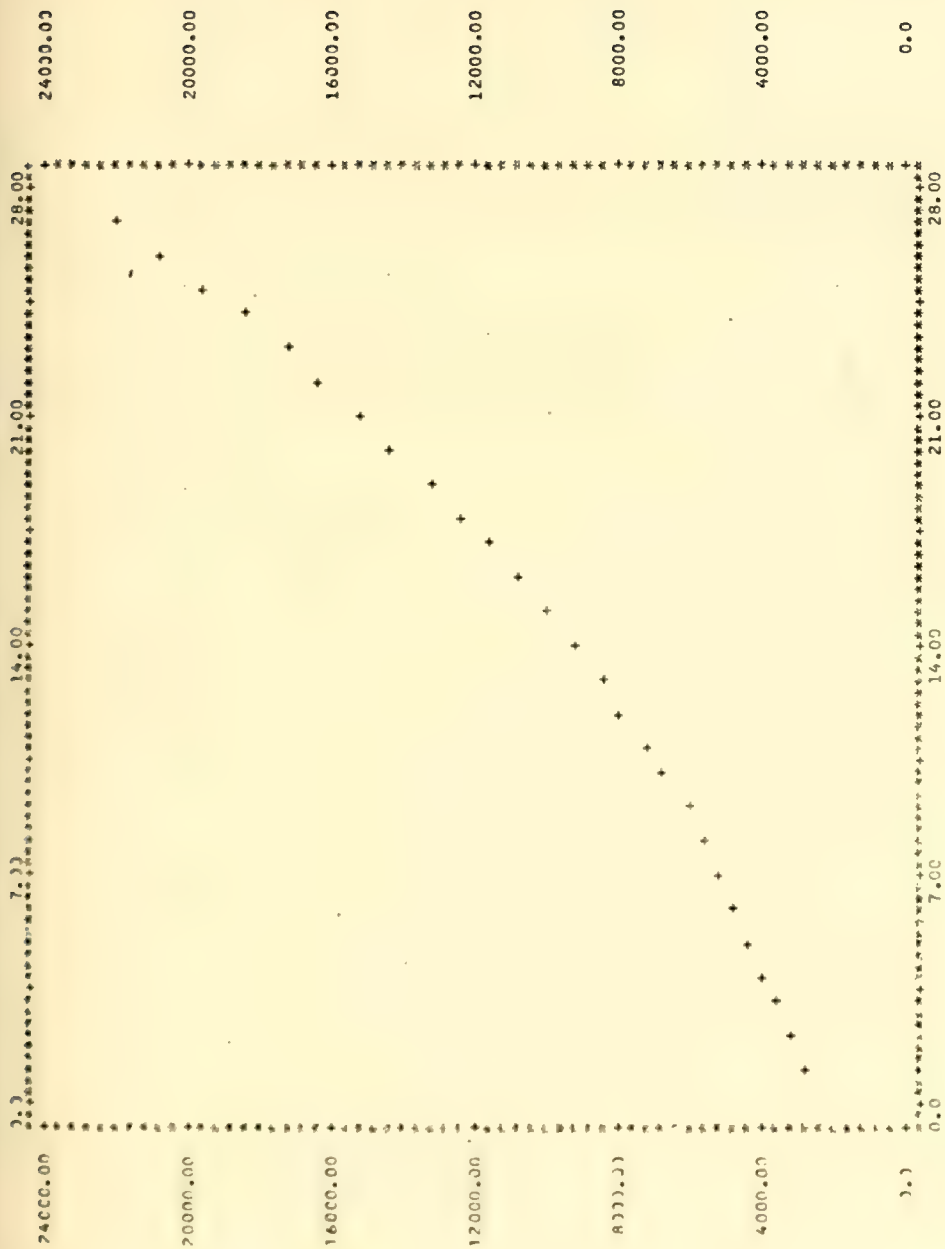
-----OMEGA BAR-----				-----LAMBDA BAR-----				-----EL-----			
( 3.100E 01	3.3	3.3	3.3	( 3.3	0.0	0.0	0.0	( 3.3	0.0	0.0	0.0
( 0.0	0.0	0.0	0.0	( 0.0	0.100E 01	0.0	0.0	( 0.0	0.0	0.0	0.0
( 0.0	0.0	0.0	0.0	( 0.0	0.0	0.100E 01	0.0	( 0.0	0.0	0.0	0.0
( 3.3	0.0	0.0	0.0	( 0.0	0.0	0.0	0.100E 01	( 0.0	0.0	0.0	0.0



STATION	RADIUS	GAMMA	OMEGA S	OMEGA THETA	DEOMEGA S	MASS
1	1.000E-03	0.000	0.000	1.000E-01	0.000	0.7330E-04
2	1.000E-03	0.000	0.000	1.000E-01	0.000	0.7612E-04
3	1.000E-03	0.000	0.000	1.000E-01	0.000	0.7894E-04
4	1.000E-03	0.000	0.000	1.000E-01	0.000	0.8176E-04
5	1.000E-03	0.000	0.000	1.000E-01	0.000	0.8458E-04
6	1.000E-03	0.000	0.000	1.000E-01	0.000	0.8740E-04
7	1.000E-03	0.000	0.000	1.000E-01	0.000	0.9022E-04
8	1.000E-03	0.000	0.000	1.000E-01	0.000	0.9303E-04
9	1.000E-03	0.000	0.000	1.000E-01	0.000	0.9585E-04
10	1.000E-03	0.000	0.000	1.000E-01	0.000	0.9867E-04
11	1.000E-03	0.000	0.000	1.000E-01	0.000	0.1015E-03
12	1.000E-03	0.000	0.000	1.000E-01	0.000	0.1043E-03
13	1.000E-03	0.000	0.000	1.000E-01	0.000	0.1071E-03
14	1.000E-03	0.000	0.000	1.000E-01	0.000	0.1099E-03
15	1.000E-03	0.000	0.000	1.000E-01	0.000	0.1128E-03
16	1.000E-03	0.000	0.000	1.000E-01	0.000	0.1156E-03
17	1.000E-03	0.000	0.000	1.000E-01	0.000	0.1184E-03
18	1.000E-03	0.000	0.000	1.000E-01	0.000	0.1212E-03
19	1.000E-03	0.000	0.000	1.000E-01	0.000	0.1240E-03
20	1.000E-03	0.000	0.000	1.000E-01	0.000	0.1267E-03
21	1.000E-03	0.000	0.000	1.000E-01	0.000	0.1295E-03
22	1.000E-03	0.000	0.000	1.000E-01	0.000	0.1323E-03
23	1.000E-03	0.000	0.000	1.000E-01	0.000	0.1351E-03
24	1.000E-03	0.000	0.000	1.000E-01	0.000	0.1379E-03
25	1.000E-03	0.000	0.000	1.000E-01	0.000	0.1407E-03
26	1.000E-03	0.000	0.000	1.000E-01	0.000	0.1435E-03
27	1.000E-03	0.000	0.000	1.000E-01	0.000	0.1463E-03

STATION	B STIFFNESS	D STIFFNESS	B PRIME	D PRIME
1	329670E	274725E	1.09890E	0.274725E
2	342350E	307658E	1.109890E	0.318616E
3	355029E	343127E	1.109890E	0.341780E
4	367709E	381228E	1.109890E	0.365757E
5	380388E	422222E	1.109890E	0.390548E
6	393068E	463563E	1.109890E	0.416151E
7	405747E	512182E	1.109890E	0.442569E
8	418427E	561714E	1.109890E	0.469783E
9	431107E	610164E	1.109890E	0.497837E
10	443787E	670154E	1.109890E	0.526635E
11	456466E	729174E	1.109890E	0.556338E
12	469145E	791763E	1.109890E	0.586838E
13	481825E	857796E	1.109890E	0.618131E
14	494505E	927728E	1.109890E	0.650234E
15	507185E	100036E	1.109890E	0.683154E
16	519864E	115427E	1.109890E	0.716889E
17	532544E	131324E	1.109890E	0.751428E
18	545223E	147224E	1.109890E	0.786295E
19	557903E	163228E	1.109890E	0.821937E
20	570583E	179338E	1.109890E	0.857937E
21	583263E	195553E	1.109890E	0.893773E
22	595943E	211863E	1.109890E	0.936340E
23	608623E	228263E	1.109890E	0.975760E
24	621303E	244763E	1.109890E	0.101570E
25	633983E	261263E	1.109890E	0.105704E
26	646663E	277763E	1.109890E	0.109890E
27	659343E	294263E	1.109890E	0.109890E





X-SCALE: "H"= 0.350E 00 UNITS  
 Y-SCALE: "H"= 0.400E 03 UNITS  
 P-STIFFNESS VS STATION

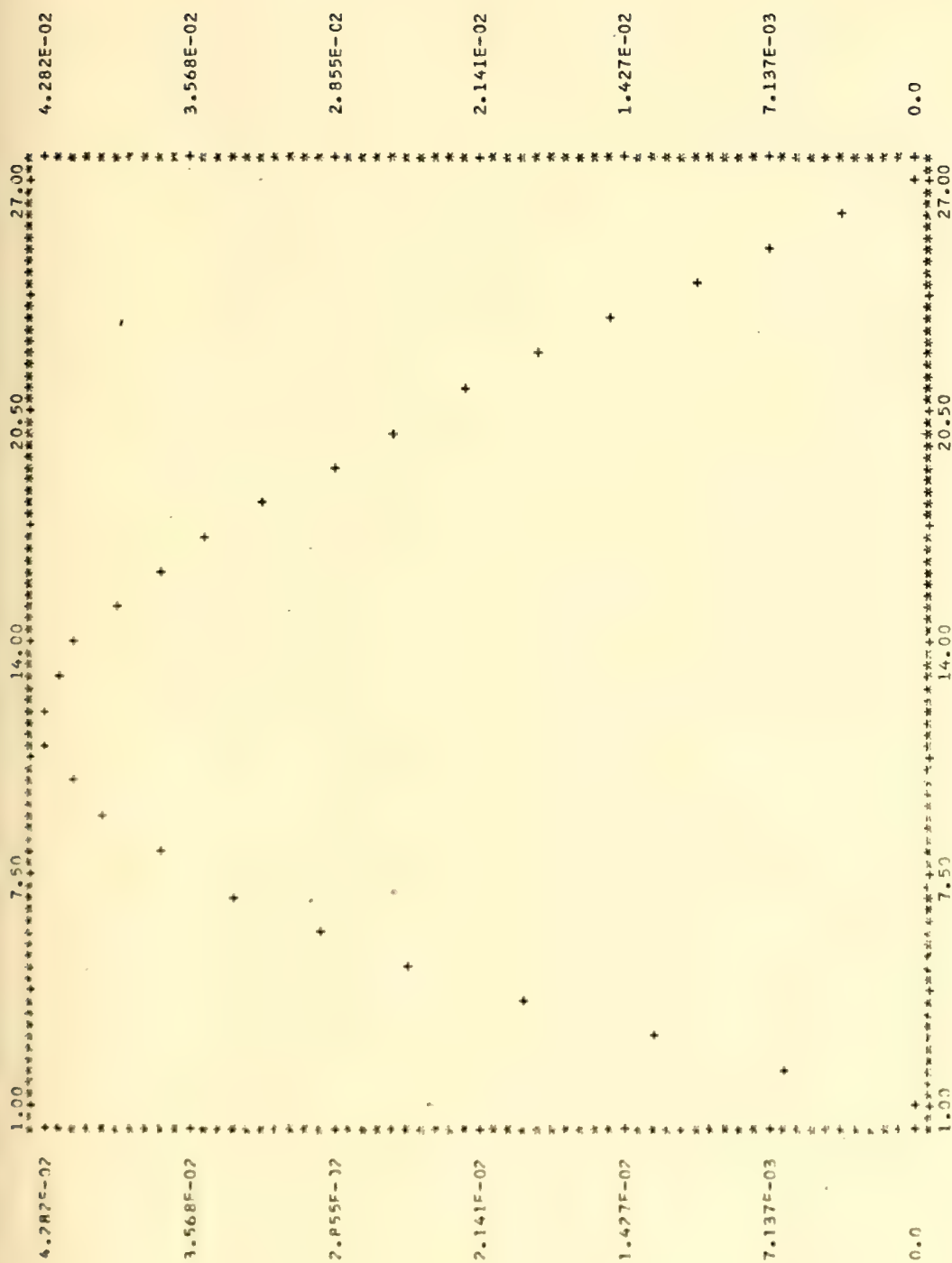
EXECUTING IN SUBROUTINE VIBERS  
 PERFORMING FREE VIBRATION ANALYSES











X-SCALE: "H"= 0.325E 0) UNITS

Y-SCALE: "H"= 0.714E-03 (UNITS)

NORMAL DEFLECTION VS STATION



[illegible]



# APPENDIX G

## EXAMPLES OF USER SUPPLIED SUBROUTINES

AND

### A LISTING OF SATANS-III

```

C THIS PROGRAM SERVES AS THE 'MAIN' PROGRAM, AND CALLS 'SATANS',
C AND CHECKS 'IRNAGN' TO THE IF WE DESIRE ANOTHER RUN
C COMMON /BLRUN/ IRNAGN
C
1 CALL SATANS
  IF (IRNAGN.EQ.1) GO TO 1
  CALL FLYNVY
  STOP
  END

```



```

SUBROUTINE GEOM
C THIS SUBROUTINE COMPUTES THE NONDIMENSIONAL
C GEOMETRY FUNCTIONS OF THE SHELL.
C
C *****
C IMPLICIT LOGICAL*1 (I)
C REAL NU,LAM,LAM2,JAY,MT,LSD18,LSD1N,MASS
C COMMON /IBL2/NN(10),MNINIT
C COMMON /IBL4/ KMAX,KL
C COMMON /BL8/ E(200),GAP(200),OMT(200)
C COMMON /BL11/ OMXI(200),PHEE,T0,T2
C COMMON /BL17/ DELMX(200)
C COMMON /BL20/ TKN,ELAST,CHAR,SIGO
C COMMON /BL32/ TTEEC,SOYINC
C COMMON /BL100/ TELDAD
C COMMON /BL102/ DELDAD
C COMMON /BL103/ MASS(200)
C CIRCULAR CYLINDRICAL SHELL WITH A LINEARLY VARYING THICKNESS
C
C LENGTH (ST) = 300.0 INCHES
C RADIUS (RAD) = 100.0 INCHES
C THICKNESS H1 = 0.1 INCHES
C H2 = 0.1 * X LP-SEC-2/INCHES**4
C DENSITY(DNSTY) = .733E-03
C
C X = 2.733E-03
C DNSTY = .1 * X
C H1 = .1
C H2 = .1
C RAD = 100.
C ST = 300.
C DEL = ST / FLOAT(KMAX - 1)
C SLOPE = (H2-H1)/ST
C DO I K
C R(K) = 1. * KMAX
C GAM(K) = RAD/CHAR
C OMT(K) = 0.0
C OMXI(K) = CHAR/RAD
C DEOMX(K) = 0.0
C S = 0.0
C H * FLCAT(K-1)
C * SLOPE + H1
C MASS(K) = (DNSTY*PI*CHAR**2)/(TKN*ELAST*TEEO**2)
C
C 1 CONTINUE
C RETURN
C END

```





```

C**
C** SUBROUTINE BDB(K,B,DB,D,DD)
C**
C** THIS SUBROUTINE COMPUTES THE NONDIMENSIONAL
C** IN-PLANE AND BENDING STIFFNESS OF THE SHELL.
C**
C** REAL NU,LAM,LAM2,JAY,MT,LSD18,LSDIN
C** COMMON /BL15/ NU,UI(10),VI(10),W1(10),V2(10),U2(10),W2(10),U3(10),
C** 1 V3(10),W3(10)
C**
C** COMMON /BL17/ DEL
C** COMMON /BL32/ TKN,ELAST,CHAR,SIGO
C**
C** B(K) = (E * H)/((1.-NU**2) * (E(0)H(0)))
C** D(K) = (E * H**3)/(12. * (1.-NU**2) * E(0)H(0)**3)
C**
C** NU
C** = 3
C** = .3E8
C**
C** X
C** = 2
C** DEL * FLOAT(K-1)
C** SLOPE
C** = 1*(X-1.)/300.
C** H
C** = SLOPE + 1
C** B
C** = (E*H)/((1.-NU**2)*ELAST*TKN)
C** LB
C** = (E*SLOPE*CHAR)/((1.-NU**2)*ELAST*TKN)
C** D
C** = (E*H**3)/((1.-NU**2)*12.*ELAST*TKN**3)
C** DD
C** = (E*(H**2)*SLOPE*CHAR)/((1.-NU**2)*4.*ELAST*TKN**3)
C**
C** RETURN
C** END

```

```

C**
C** SUBROUTINE PLOAD(K)
C**
C** THIS SUBROUTINE ESTABLISHES THE NON-DIMENSIONAL FOURIER
C** COEFFICIENTS OF THE LOADS APPLIED TO THE SHELL
C**
C** COMMON /IBL1/ MNMAX
C** COMMON /IBL2/ NN(10),MNINIT
C** COMMON /IBL4/ KMAX,KL
C** COMMON /IBL8/ LSTEP,ITR
C** COMMON /BL3/ PR(10),PX(10),PT(10)
C** COMMON /BL6/ SCE,OSE,ALCAD
C** COMMON /BL8/ R(200),GAM(200),OMT(200)
C** COMMON /BL32/ TKN,ELAST,CHAR,SIGO
C** COMMON /BL102/ DELCAD
C** COMMON /BL103/ MASS(200)
C**
C** RETURN
C** END

```



```

SUBROUTINE TLOAD(K)
C THIS SUBROUTINE DESCRIBES THE THERMAL LOADING ON THE SHELL
C
REAL NU
COMMON /IBL1/ MNMAX
COMMON /IBL2/ NN(10), MNINIT
COMMON /IBL3/ NSTEP, ITR
COMMON /IBL4/ TT(10), EMT(10), DMT(10)
COMMON /IBL5/ SOE, DSE, ALPCAD
COMMON /IBL6/ NU, U1(10), V1(10), W1(10), V2(10), W2(10), U3(10),
1 V3(10), W3(10)
COMMON /IBL32/ TKN, ELAST, CHAR, SIGC
RETURN
END

```

```

SUBROUTINE INITL
C THIS SUBROUTINE DESCRIBES THE INITIAL CONDITIONS FOR DYNAMIC CASES
C
      IMPLICIT LOGICAL*1 (5)
      COMMON /IBL1/ MJMAX
      COMMON /IBL2/ NN(10), MNINIT
      COMMON /IBL4/ KMAX, KL
      COMMON /IBL9/ MAXM
      COMMON /IBL12/ KMAX1, KMAX2, NCONV
      COMMON /IBL6/ SOE, AL, AD
      COMMON /IBL32/ TKN, ELAST, CHAR, SIGO
      COMMON /IBL100/ TEEC, $DYMC
      COMMON /IBL101/ DELSD
      COMMON /IBL104/ Z(4,220), ZDOT(4,220), X(4,200), Z0(4,220), Z2(4,220),
      1 Z3(4,220)
      RETURN
END

```







SATANS-III LISTING

THE SUBROUTINES ARE LISTED, READING ACROSS, AS FOLLOWS:

NAME	CARD NUMBER	NAME	CARD NUMBER	NAME	CARD NUMBER	NAME	CARD NUMBER
SATANS	10	STATIC	2540	DYNMAC	5610	VIBERS	8680
PMATRX	9240	XANDZ	11560	FORCE	14270	HJ	15890
FFG	16610	ABC	17670	PANDC	18000	MODES	18370
POLE	19360	INLPOL	20880	FNLPOL	21560	PHIBET	22210
TEAETA	23110	UPDATE	24140	OUTPUT	24520	PLOT1	28340
PLOT2	29410	PLOTIT	31380	SCALIT	31950	ROUND	32740
DRAWIT	33220	RANGER	34720	ADJUST	35540	BTTM	36590
TOPPER	37370	ITREFL	38050	ITRATE	38520	ITCPLX	41130
INITAA	42000	VEGOUT	43060	START	44780	MATINV	45150
		FLYINVY	46560				

```

SUBROUTINE SATANS
C THIS SUBROUTINE READS ALL DATA, PRINTS THE OUTPUT TITLE PAGE,
C AND PASSES CONTROL TO ONE OF THE MAJOR CONTROLLING SUBROUTINES
C
C IMPLICIT LOGICAL*1 ($)
REAL*8 EPSVIB, EPSVEC, EPSAIT
REAL*4 NU, LAM, LAM2, JAY, NT, LSD18, LSDIN, MASS
COMMON /IBL1/ MNMAX
COMMON /IBL4/ KMAX, KL
COMMON /IBL5/ IBCINL, IBCENL
COMMON /IBL7/ MNMAXC, MAXD(10), MAXS(10), MAXSY(10), IS(10,10),
1 JS(10,10), ID(10,10), JD(10,10), IJS(10)
COMMON /IBL9/ MAXM
COMMON /IBL10/ IFREQ, NTHMAX

```

10  
20  
30  
40  
50  
60  
70  
80  
90  
100  
110  
120  
130  
140  
150









```

C** IF($VIBES) CALL INITAA
C** THIS CONCLUDES DATA READ-IN FOR THIS RUN.
C** PRINT OUT PERTINENT DATA.
C**
C** WRITE(6,200) NO
C** WRITE(6,201) TITLE
C** IF (IIRCINL.LT.0) WRITE(6,203)
C** IF (IIRCINL.LT.0) GO TO 3
C** WRITE(6,204)
C** WRITE(6,205) (OMEG1(1,J),J=1,4),(CAPL1(1,J),J=1,4),ELL(1)
C** WRITE(6,206) (OMEG1(2,J),J=1,4),(CAPL1(2,J),J=1,4),ELL(2)
C** WRITE(6,207) (OMEG1(3,J),J=1,4),(CAPL1(3,J),J=1,4),ELL(3)
C** WRITE(6,208) (OMEG1(4,J),J=1,4),(CAPL1(4,J),J=1,4),ELL(4)
C** IF (IIRCFNL.LT.0) WRITE(6,209)
C** IF (IIRCFNL.LT.0) GO TO 4
C** IF (IIRCFNL.LT.0) GO TO 4
C** WRITE(6,211)
C** WRITE(6,210) (OMEG1(1,J),J=1,4),(CAPL1(1,J),J=1,4),ELL(1)
C** WRITE(6,205) (OMEG1(2,J),J=1,4),(CAPL1(2,J),J=1,4),ELL(2)
C** WRITE(6,206) (OMEG1(3,J),J=1,4),(CAPL1(3,J),J=1,4),ELL(3)
C** WRITE(6,207) (OMEG1(4,J),J=1,4),(CAPL1(4,J),J=1,4),ELL(4)
C** WRITE(6,211)
C** IF ($VIBES) GO TO 11
C** IF ($DYNMC) GO TO 5
C** IF (6,212) KMAX,MAX,DELGAD,LSHAX,ITRMAX,LCHMAX,EPS
C** WRITE(6,213) CHAR,TKN,PLAST,SIGCO,TEEO,NU
C** GO TO 6
C** WRITE(6,246) KMAX,MAX,ITRVAX,LSMAX,DELGAD,EPS
C** WRITE(6,247) CHAR,TKN,PLAST,SIGCO,TEEO,NU
C** GO TO 6
C** WRITE(6,252) KMAX,ITP1,VVIBES,EPS
C** WRITE(6,247) CHAR,TKN,PLAST,SIGCO,TEEO,NU
C** IF ($PLCOTS) GO TO 9
C** IF (NOT) ICHCK1=IABS(IRADII)+IABS(OMEGS)+IABS(OMEGT)
C** +IABS(IDECMS)+IABS(IPR)+IABS(IDSTIF)+IABS(IPS)+IABS(IPT)+IABS(ITT)
C** +IABS(IDDSTF)+IABS(IDTT)+IABS(IDMT)
C** +IABS(INT)+IABS(1D14)
C** ICHCK1.GT.0) WRITE(6,214)
C** ICHCK1.FO.0) GO TO 8
C** IF (IRADII.NE.0) WRITE(6,215)
C** IF (IGAMMA.NE.0) WRITE(6,216)
C** IF (IGAMMA.NE.0) WRITE(6,217)
C** IF (OMEGS.NE.0) WRITE(6,218)
C** IF (IDECMS.NE.0) WRITE(6,219)
C** IF (IDSTIF.NE.0) WRITE(6,220)
C** IF (IDSTIF.NE.0) WRITE(6,221)

```









```

107 FORMAT (I2)
108 FORMAT (I5,4F1.0,F16.3,15)
109 FORMAT (3I5,3F15.8)
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
```









```

SUBROUTINE STATIC
C*** THIS SUBROUTINE IS THE MAJOR CONTROLLING ROUTINE FOR ALL STATIC ***
C*** ANALYSIS PROBLEMS. IT CONDUCTS INITIALIZATION, PREPARES INPUT ***
C*** MATERIAL FOR PLOTTING IF REQUIRED. DETERMINES SHELL GEOMETRY, STIFF- ***
C*** NESSSES, LOADING (PHYSICAL AND/OR THERMAL), AND INITIAL CONDITIONS. ***
C*** IT CONTROLS PROBLEM SOLUTION PROCEDURE. ***
C*** IMPLICIT LOGICAL*1 ($) ***
C*** REAL*4 NU,LAM,LAM2,JAY,MT, LSD18,LSDIN,MASS ***
COMMON /IBL1/ NMAX
COMMON /IBL2/ 'N' APPEARS AS 'NN' IN SUBROUTINES PLOAD & EFG
C
COMMON /IBL3/ NO,M1,M2,M3
COMMON /IBL4/ KMAX,KL
COMMON /IBL5/ IBCINL,IBCENL
COMMON /IBL6/ KLL
COMMON /IBL7/ NMAXO,MAXD(10),MAXS(10),MAXSY(10),IS(10,10),
1 JS(10,10),ID(10,10),JD(10,10),IJS(10,
COMMON /IBL8/ LSTEP,IIR
COMMON /IBL9/ MAXM
COMMON /IBL10/ IFRFEC,NIHVAX
COMMON /IBL11/ ICCRFL,IIPASS
COMMON /IBL12/ KMAX1,KMAX2,NCONV
COMMON /IBL13/ ITRMAX,LMAX
COMMON /IBL1/ A(4,4),B(4,4),C(4,4)
COMMON /BL1/ PR(10),PX(10),PT(10)
COMMON /BL3/ P(4,4,200),ZFIM(4,4,10),ZF2M(4,4,10),ZF3M(4,4,10),
COMMON /BL4/ ZF4M(4,4,10),DT(10),DMT(10)
1 COMMON /BL5/ 'MT' APPEARS AS 'ERT' IN SUBROUTINES INLPOL & FNLPOL
C
COMMON /BL6/ SOE,CSE,ALCAD
COMMON /BL7/ DI,S1
COMMON /BL8/ R(200),GAM(200),DMT(200)
COMMON /BL9/ RFS(4,10),ELIS(4),CEES(4,10)
COMMON /BL10/ PHIX(10),PHIT(10),PHI(10)
COMMON /BL11/ CMXI(200),PHEE,T0,T2
COMMON /BL12/ TDLI,TCEL
COMMON /BL13/ OMEGI(4,4),CAPL1(4,4),OMEGL(4,4),CAPLL(4,4),
1 COMMON /BL14/ UNIT(4,4)
COMMON /BL15/ LAM2,LSD18,LSDIN
COMMON /BL15/ NU,U1(10),V1(10),W1(10),V2(10),U2(10),W2(10),U3(10),
1 V3(10),W3(10)
COMMON /BL16/ EPS
COMMON /BL17/ DEL
COMMON /BL18/ DEL1(4),ELL(4)
COMMON /BL19/ TH(36)
COMMON /BL20/ DECMX(200)

```









```

DO 98 J=1,4
KKLM=J/4+1
OMGL(I,J)=OMGL(I,J)*SIGT(KKLM)
CAPLI(I,J)=CAPLI(I,J)*SIGC(KKLM)
14 IF (IBCFNL.LT.0) GO TO 17
DO 99 I=1,4
CC=99 J=1,4
KKLM=J/4+1
OMGL(I,J)=OMGL(I,J)*SIGT(KKLM)
CAPLI(I,J)=CAPLI(I,J)*SIGC(KKLM)
17 LAM=TKN/CHAR
SOF=SIGD/ELAST
CSE=.5*SCE
CI=1.0-NU
SI=1.0+NU
LM2=LAM*2
1F(NDIMEN.LT.1) GO TO 228
SIGC=1.0
ELAST=1.0
TKN=1.0
CHAR=1.0
DO 230 M=1,MAXM
N(M)=0.0
PX(M)=0.0
PT(M)=0.0
PR(M)=0.0
TT(M)=0.0
MT(M)=0.0
DMT(M)=0.0
MAXD(M)=0
MAXS(M)=0
MAXSY(M)=0
CALL GFCM
1 ICHK1=IABS(IGAMMA)+IABS(IOMEGS)+IABS(IOMEGT)+IABS(IDEOMS)
+IABS(IRADII)
1 ICHK2=IABS(IBSTIF)+IABS(IDSTIF)+IABS(IBBSTF)+IABS(DDSTF)
IF (.NOT.$PLOTS) GO TO 11
DO 2 K=1,KMAX
XSTATN(K)=FLOCAT(K)
2 IF (ICCHK1.EQ.0) GO TO 11
DO 1 K=1,KMAX
XRADII(K)=R(K)*CHAR
YGAMMA(K)=GAM(K)/CHAR
YOMEGS(K)=CMXI(K)/CHAR
YOMEGT(K)=DMT(K)/CHAR
YDEOMS(K)=DEOMX(K)/(CHAR*CHAR)
1 CONTINUE

```





```

11 CONTINUE
86 DO 86 K=1,KMAX
   MASS(K)=0.
   WRITE(6,802)
   DO 978 K=1,KMAX
     RKK=R(K)*CHAR
     CMXIK=CMXI(K)/CHAR
     GAMK=GAM(K)/CHAR
     CMTK=CMT(K)/CHAR
     DEOMXK=DEOMX(K)/(CHAR*CHAR)
978   WRITE(6,803) K,RKK,GAMK,CMXIK,CMTK,DEOMXK
805   MO=0
      M1=0
      M2=0
      M3=0
      ABN=CHAR/SIGC/TKN
      ZN=SIGC*TKN
      WRITE(6,112)
      DO 888 K=1,KMAX
        CALL BDB(K,B,DB,D,DD)
        RST=ELAST*TKN
        ZST=ELAST*TKN*3
        B=B*BST
        C=C*ZST
        DB=DB/CHAR*BST
        CC=OD/CHAR*ZST
      WRITE(6,71) K,B,D,CB,DD
      IF (.NOT.$PLOTS.CR.(ICHECK2.EQ.0)) GO TO 888
      YBSTIF(K)=B
      YCBSTIF(K)=DB
      YCDSTIF(K)=DD
888   CCNTINUE
      CALL PLOAD(1)
      CALL TLOAD(1)
      DO 889 M=1,MNMAX
        WRITE(6,113) N(M)
        WRITE(6,114)
          ICHK3=IABS(IPR)+IABS(IPT)+IABS(ITT)+IABS(IMT)
          +IABS(IDTT)+IABS(IDMT)
1      DO 890 K=1,KMAX
        CALL PLOAD(K)
        CALL TLOAD(K)
        PRM=PR(M)/ABN
        PTM=PT(M)/ABN
        PXM=PX(M)/ABN
        TIM=TT(M)*ZN
        EMTM=MT(M)/CHAR*ZN*TKN

```



```

DTM=DT(M)/CHAR*ZN
DMTM=DMT(M)*ZN*TKN*(CHAR*CHAR)
WRITE (6,115) K,PRM,PXM,PTM,TTM,EMTM,DTM,DMTM
IF (.NOT.$PLOTS.OR.(ICHCK3.EQ.0)) GO TO 890
YPS(K)=PRM
YPS(K)=PXM
YPT(K)=PTM
YTT(K)=TTM
YMT(K)=EMTM
YCTT(K)=DTM
YCMT(K)=DMTM
CONTINUE
890 IF (M.EQ.1) ICHCK3=ICHCK1+ICHCK2+ICHCK3
IF ($PLOTS.AND.(ICHCK3.GT.0)) CALL PLOT1(M)
CONTINUE
DELSQ=DELL*#2
TDLI=.5/DEL
TDEL=2.0*DEL
MNINIT=1
MNXAXO=MNMAX
DO 20 I=1,4
DO 20 J=1,4
UNIT(I,J)=0.0
UNIT(I,J)=1.0
20 IF (I.EQ.J) UNIT(I,J)=1.0
NMAX=MAXM*KMAX2
DO 22 K=1,NMAX
DO 22 I=1,4
ZCOT(I,K)=0.0
ZC(I,K)=0.0
Z2(I,K)=0.0
Z3(I,K)=0.0
Z(I,K)=0.0
22 ALOAD=DELOAD
CALL PMATRX
LSTEP=1
LCHANG=0
ICORFL=0
IF (MNMAX.EQ.MAXM) ICORFL=1
IPASS=0
CALL XANDZ
IF (ITRMAX.EQ.1) GO TO 50
NMNAXO=MNMAX
IF (IPASS.LT.2) CALL MODES
IF (INC.NV.EQ.1).AND.(ITR.GT.1) GO TO 50
IF (ITR.LT.ITRMAX) GO TO 23
IF (LCHANG.LT.LCHMAX) GO TO 30
WRITE (6,220) NO

```

4460  
4470  
4480  
4490  
4500  
4510  
4520  
4530  
4540  
4550  
4560  
4570  
4580  
4590  
4600  
4610  
4620  
4630  
4640  
4650  
4660  
4670  
4680  
4690  
4700  
4710  
4720  
4730  
4740  
4750  
4760  
4770  
4780  
4790  
4800  
4810  
4820  
4830  
4840  
4850  
4860  
4870  
4880  
4890  
4900  
4910  
4920  
4930



```

4940 GO TO 500
4950 FL=LSTEP
4960 LI=IPRINT
4970 FLI=LI
4980 FI=FLI-FL/FI
4990 IF(FT.EQ.0.) CALL OUTPUT(IMODE)
5000 IF(LSTEP.EQ.1) ITR=1
5010 IF(LSTEP.EQ.1) ITRPR=1
5020 IF(ITR.GT.ITRPR) ITRPR=ITR
5030 IF(LSTEP.GE.LSMAX) GO TO 360
5040 DO 61 MN=1,MNMAX
5050 DO 61 K=1,KMAX2
5060 IK=K+(MN-1)*KMAX2
5070 DO 61 I=1,4
5080 ZN=2.0-Z(I,IK)-ZO(I,IK)
5090 ZC(I,IK)=Z(I,IK)
5100 Z(I,IK)=ZN
5110 IF(LSTEP.GE.LSMAX) GO TO 360
5120 ALOAD=ALOAD+DELOAD
5130 LSTEP=LSTEP+1
5140 ITR=1
5150 GO TO 400
5160 WRITE(6,221) NO
5170 GO TO 500
5180 ITR=ITR+1
5190 GO TO 400
5200 IF(LSTEP-1) 310,310,320
5210 WRITE(6,223)
5220 GO TO 500
5230 WRITE(6,222)
5240 LCHANG=LCHANG+1
5250 LSTEP=LSTEP-1
5260 ALOAD=ALOAD-DELOAD
5270 DELOAD=DELOAD/5.0
5280 DO 32 MN=1,MNMAX
5290 DO 32 K=1,KMAX2
5300 IK=K+(MN-1)*KMAX2
5310 DO 32 I=1,4
5320 ZC(I,IK)=ZO(I,IK)
5330 GO TO 62
5340 C*****
5350 71 FORMAT(20X,13,4X,4E20,6)
5360 112 FORMAT(///17X,12H STATION 20H B STIFFNESS 20H D ST
5370 113 IFNESS 20H D PRIME //)
5380 113 FORMAT(///25X,44HPRESSURE AND TEMPERATURE COEFFICIENTS FOR N=13,8H
5390 1 FOLLOW//)
5400 114 FORMAT(5X,7HSTATION,3X,15H PX 15H

```



5420  
5430  
5440  
5450  
5460  
5470  
5480  
5490  
5500  
5510  
5520  
5530  
5540  
5550  
5560  
5570  
5580  
5590  
5600

```

1 PT      15H      TT      MT      15H      DTT      1
2 5H      DMT      //)
115 FORMAT(6X,13.7X,7E15.4) THE MAXIMUM NUMBER OF LOAD CHANGES HAVE BE
220 FORMAT(1H1,80H END PROBLEM NUMBER I4) THE MAXIMUM NUMBER OF LOAD STEPS HAVE BEEN
221 FORMAT(1H1,79H END PROBLEM NUMBER I4) THE MAXIMUM NUMBER OF LOAD STEPS HAVE BEEN
222 1 TAKEN. END PROBLEM NUMBER I4) THE SOLUTION DID NOT CONVERGE WITHIN THE M
223 1 MAXIMUM NUMBER OF ITERATIONS. THE LOAD FACTOR HAS BEEN DIVIDED BY
25.)
223 FORMAT(1H1,69H THE SOLUTION DID NOT CONVERGE FOR THE FIRS
224 1T LOAD INCREMENT.//11X,71HLOOK FOR AN ERROR IN THE INPUT DATA, OR T
225 2RY A SMALLER VALUE FOR DELTA.)
802 FORMAT(1H1,17X,15H STATION 16H RADIUS 16H DEOMEGA S //)
803 16H OMEGA S 16X,5E16.4)
8888 FORMAT ('0',120,'EXECUTING IN SUBROUTINE "STATIC"')
500 RETURN
END

```

5610  
5620  
5630  
5640  
5650  
5660  
5670  
5680  
5690  
5700  
5710  
5720  
5730  
5740  
5750  
5760  
5770  
5780  
5790  
5800  
5810  
5820  
5830  
5840  
5850  
5860  
5870

```

SUBROUTINE DYNAMIC
C THIS SUBROUTINE IS ONE OF THE MAJOR CONTROLLING SUBROUTINES FOR
C ALL DYNAMIC ANALYSIS PROBLEMS. IT OPERATES IN A FASHION SIMILAR
C TO SUBROUTINE STATIC.
C IMPLICIT LOGICAL*1 ($)
C REAL*4 NU,LAM,LAM2,JAY,MT,LSD18,LSD1N,MASS
C COMMON /IBL1/ MNMAX
C COMMON /IBL2/ N(10),MNIN,IT
C COMMON /IBL3/ MO,M1,M2,M3
C COMMON /IBL4/ KMAX,KL
C COMMON /IBL5/ IBCINL,IBCENL
C COMMON /IBL6/ KLL
C COMMON /IBL7/ MNMAXO,MXO(10),MAXS(10),MAXSY(10),IS(10,10),
1 JS(10,10),ID(10,10),JD(10,10),IJS(10)
C COMMON /IBL8/ LSTEP,IT
C COMMON /IBL9/ MAXM
C COMMON /IBL10/ IPRREC,NTRMAX
C COMMON /IBL11/ IPRREL,IPRASS
C COMMON /IBL12/ KMAX1,KMAX2,NCONV
C COMMON /IBL13/ ITRMAX,LSMAX
C COMMON /IBL14/ A(4,4),BE(4,4),C(4,4)
C COMMON /IBL15/ PR(10),PK(10),PT(10)
C COMMON /IBL16/ P(4,4,200),ZF1M(4,4,10),ZF2M(4,4,10),ZF3M(4,4,10),
1 ZF4M(4,4,10)

```









```

4      YNSTH(100),YQS(100),YMS(100),YMTN(100),YMSTH(100),
5      YU(100),YV(100),YW(100),YPHIS(100),YPHIT(100),
6      YPHI(100),XSTATN(100)
COMMON /BLK1/ VMAS,NN2,$VIBES
DIMENSION SIGT(2),SIGC(2),TITLE(18)
DIMENSION VBAR(10),AVR(10)
WRITE (6,8888)
DELSO=DELOAD#DELOAD
KL=KMAX-1
KLL=KMAX-2
KMAX1=KMAX+1
KMAX2=KMAX+2
AK=KL
SIGT(1)=SIGC*TKN
SIGT(2)=SIGC/ELAST
SIGC(1)=SIGC*CHAR/ELAST
SIGC(2)=SIGC*TKN**3/CHAR
IF (18CINL.LT.0) GO TO 14
DO 98 I=1,4
DO 98 J=1,4
KKLM=J/4+1
CMEGL(I,J)=OMEG(I,J)*SIGT(KKLM)
CAPL(I,J)=CAPL(I,J)*SIGC(KKLM)
98      IF (18CFNL.LT.0) GO TO 17
DO 99 I=1,4
DO 99 J=1,4
KKLM=J/4+1
CMEGL(I,J)=CMEGL(I,J)*SIGT(KKLM)
CAPL(I,J)=CAPL(I,J)*SIGC(KKLM)
99      LAM=TKN/CHAR
SCE=SIGC/ELAST
OSF=.5#SCE
OI=1.0-NU
SI=1.0+NU
LAM2=LAM**2
IF (NDIMEN.LT.1) GO TO 228
SIGC=1.0
ELAST=1.0
TKN=1.0
CHAR=1.0
DO 230 M=1,MAXM
N(M)=0.0
PX(M)=0.0
PT(M)=0.0
PR(M)=0.0
TT(M)=0.0
228

```



```

6840 MT(M)=0.0
6850 CT(M)=0.0
6860 DMT(M)=0.0
6870 MAXD(M)=0
6880 MAXS(M)=0
6890 MAXSY(M)=0
6900 CALL GEOM
6910 ICHCK1=IABS(IGAMMA)+IABS(IOMEGS)+IABS(IOMEGT)+IABS(IDEOMS)
6920 ICHCK2=IABS(IRADII)+IABS(IGAMMA)+IABS(IOMEGS)+IABS(IOMEGT)+IABS(IDEOMS)
6930 ICHCK2=IABS(IRSTIF)+IABS(IDSTIF)+IABS(IBBSTF)+IABS(IDDSTF)
6940 IF (.NOT. $PLOTS) GO TO 11
6950 DO 2 K=1,KMAX
6960 XSTATN(K)=FLOAT(K)
6970 IF (ICHCK1.EQ.0) GO TO 11
6980 DO 1 K=1,KMAX
6990 XRADII(K)=R(K)*CHAR
7000 YGAMMA(K)=GAM(K)/CHAR
7010 YOMEGS(K)=CMXI(K)/CHAR
7020 YOMEGT(K)=CMT(K)/CHAR
7030 YDECHS(K)=DEOMX(K)/(CHAR*CHAR)
7040 CONTINUE
7050 ICHCK1=0
7060 WRITE(6,810)
7070 IF (NDIMEN.EQ.1) TEED=1.0
7080 DO 979 K=1,KMAX
7090 RKK=R(K)*CHAR
7100 CMXIK=CMXI(K)/CHAR
7110 GAMK=GAM(K)/CHAR
7120 CMTK=CMT(K)/CHAR
7130 DEOMXK=DEOMX(K)/(CHAR*CHAR)
7140 AMSS=MASS(K)*TEED*TKN/CHAR**2
7150 WRITE(6,813) K,RKK,GAMK,CMXIK,DEOMXK,AMSS
7160
7170
7180
7190
7200
7210
7220
7230
7240
7250
7260
7270
7280
7290
7300
7310

```



```

WRITE (6,71) K,B,D,DB,DD
IF (.NOT.$PLOTS.OR.(ICHECK2.EQ.0)) GO TO 888
YBSTIF(K)=8
YDSTIF(K)=D
YB8STIF(K)=DB
YDDSTIF(K)=DD
CCNT INUE
CALL PLOAD(1)
CALL TLOAD(1)
DELSO=DEL*2
TDLI=.5/DEL
TDEL=2.0*DEL
MUNIT=1
MNMNAXO=MNMNAX
DO 20 I=1,4
DO 20 J=1,4
UNIT(I,J)=0.0
IF(I.EQ.J) UNIT(I,J)=1.0
NMAX=MAXM*KMAX2
ICHECK3.=ICHECK1 + ICHECK2
IF($PLOTS.AND.(ICHECK3.GT.0)) CALL PLOT1(1)
IF($VIBES) CALL VIBERS(8500)
DO 22 K=1,NMAX
DO 22 I=1,4
ZDOT(I,K)=0.0
ZQ(I,K)=0.0
Z2(I,K)=0.0
Z3(I,K)=0.0
Z4(I,K)=0.0
IF(IC.EQ.0) GO TO 834
CALL INITL
ACQ=CHAR*SIGO/ELAST
ACM=SIGO*TKN*.3/CHAR
DO 830 M=1,MNMAX
NM=(M-1)*KMAX2
WRITE(6,126) N(M)
WRITE(6,127)
DO 831 K=2,KMAX1
MK=K+MM
TU=ACQ*ZQ(1,MK)
TV=ACQ*ZQ(2,MK)
TW=ACQ*ZQ(3,MK)
TM=ACM*ZQ(4,MK)
KK=K-1
831 WRITE(6,71) KK,TU,TV,TW,TM
WRITE(6,129)
DO 830 K=2,KMAX1
ACD=CHAR*SIGO/(ELAST*TM=EO)

```

7320  
7330  
7340  
7350  
7360  
7370  
7380  
7390  
7400  
7410  
7420  
7430  
7440  
7450  
7460  
7470  
7480  
7490  
7500  
7510  
7520  
7530  
7540  
7550  
7560  
7570  
7580  
7590  
7600  
7610  
7620  
7630  
7640  
7650  
7660  
7670  
7680  
7690  
7700  
7710  
7720  
7730  
7740  
7750  
7760  
7770  
7780  
7790





```

AMD=SIGO*TKN**3/(CHAR*TEEO)
MK=K+MM
TU=ACD*ZDOT(1,MK)
TV=ACD*ZDOT(2,MK)
TW=ACD*ZDOT(3,MK)
TM=AMD*ZDOT(4,MK)
KK=K-1
WRITE(6,71) KK,TU,TV,TW,TM
DO 830 I=1,4
  Z(I,MK)=ZO(I,MK)+ZDCT(I,MK)*DELOAD
  Z2(I,MK)=ZO(I,MK)-ZDCT(I,MK)*DELOAD
  Z3(I,MK)=ZO(I,MK)-2.*ZDCT(I,MK)*DELOAD
830 CCNTINUE
834 ALCAD=1.0
CALL PMATRX
LSTEP=1
LCHANG=0
ITR=1
ICORFL=J
IF(MNMAX.EQ.MAXM) ICORFL=1
IPASS=0
CALL XANDZ
IF(ITRMAX.EQ.1) GO TO 50
MMAXO=MNMAX
IF(IPASS.LT.2) CALL MODES
IF(NCONV.EQ.1) GO TO 5C
IF(ITR.LT.ITRMAX) GC TO 23
GO TO 365
50 FL=LSTEP
FI=IPRINT
LI=LSTEP/IPRINT
FLI=LI-FL/FI
FT=FLI-FL/FI
IF(FT.EQ.0.) CALL OUTPUT(IMODE)
IF(LSTEP.EQ.1) ITR=1
IF(LSTEP.EQ.1) ITRPR=1
IF(ITR.GT.ITRPR) ITRPR=ITR
IF(LSTEP.GE.LSMAX) GO TO 360
DO 65 MN=1,MNMAX
DO 65 K=1,KMAX2
IK=K+(MN-1)*KMAX2
DO 65 I=1,4
  ZN=3.0*(Z(I,IK)-ZO(I,IK))+Z2(I,IK)
  Z3(I,IK)=Z2(I,IK)
  Z2(I,IK)=ZO(I,IK)
  ZO(I,IK)=Z(I,IK)
  Z(I,IK)=ZN
  ALCAD=1.0
65

```



8280  
8290  
8300  
8310  
8320  
8330  
8340  
8350  
8360  
8370  
8380  
8390  
8400  
8410  
8420  
8430  
8440  
8450  
8460  
8470  
8480  
8490  
8500  
8510  
8520  
8530  
8540  
8550  
8560  
8570  
8580  
8590  
8600  
8610  
8620  
8630  
8640  
8650  
8660  
8670

```

LSTEP=LSTEP+1
ITR=1
GO TO 400
ITR=ITR+1
23 GO TO 400
360 WRITE(6,271) (AVB(M),M=1,MAXM)
WRITE(6,188) ITRPR
GO TO 500
365 IF(LSTEP.EQ.1) GO TO 367
WRITE(6,266) ITRMAX,LSTEP,NQ
WRITE(6,188) (AVB(M),M=1,MAXM)
GO TO 500
367 WRITE(6,273)
GO TO 500
C *****
71 FORMAT(20X,I3,4X,4E2C,6)
112 FORMAT(///,17X,12H STATION 20H B STIFFNESS 20H D ST
113 IF STIFFNESS 20H D PRIME //)
126 FORMAT(///,5X,29H THE INITIAL CONDITIONS FOR N=I3,8H FOLLOW//)
127 FORMAT(19X,7H STATION,3X,20H U M S DOT 20H V DOT
129 FORMAT(///,15X,7H STATION,3X,20H W U DOT M S DOT 20H V DOT
188 FORMAT(///, THE MAXIMUM VBAR FOR EACH MODE IS,10E11,4)
189 FORMAT(///, THE MAXIMUM NUMBER OF ITERATIONS TAKEN IS ,14)
266 FORMAT(1H,35H THE SOLUTION DID NOT CONVERGE IN I3,24H ITERATIONS
271 AT TIME STEPS 5,21H. END PROBLEM NUMBER I4,1H.)
271 FORMAT(1H,79H THE MAXIMUM NUMBER OF TIME STEPS HAVE BEEN
273 TAKEN. END PROBLEM NUMBER I4)
273 FORMAT(1H,69H THE SOLUTION DID NOT CONVERGE FOR THE FIRST
IT TIME INCREMENT,11X,71H LOAD.)
2RY A SMALLER VALUE FOR DELTA.
810 FORMAT(1H,5X,15H STATION 16H RADIUS 16H GAMMA
16H DEOMEGA S 16H
2 MASS //)
813 FORMAT(8X,I3,9X,6E16,6)
8683 FORMAT(10,120,EXECUTING IN SUBROUTINE "DYNAMIC")
500 RETURN
END

```

8680  
8690  
8700  
8710  
8720  
8730

```

SUBROUTINE VIBERS(4)
C *****
C THIS IS THE MAJOR CONTROLLING SUBROUTINE FOR THE FREE VIBRATION
C ANALYSIS PORTION OF THE PROGRAM. THE MODE OF OPERATION (FREQUENCY
C RANGE SEARCH, SUPPLIED POINTS, MINIMUM EIGENVALUE) IS
C DETERMINED AND APPROPRIATE SUBROUTINES CALLED.

```



```

C *****
IMPLICIT LOGICAL=1 (S)
REAL*8 EPSVIB, EPSVEC, EPSAIT, TWOPI, WV, XV, YV
COMMON /IBL2/ N(10), MN, INIT
COMMON /IBL7/ MNMAXO, MAXO(10), MAXS(10), MAXSY(10), MODVIB(3,100), ITEM
1P(110)
COMMON /BL100/ TEEC, $DYNMC
COMMON /BL101/ DELSD
COMMON /BL104/ WV(4,203), XV(4,203), YV(4,203), FILL(28), SHFTPT(3,100)
COMMON /BLK2/ IVIBES, IVB
COMMON /BLK3/ TWOPI, IBEGIN, IEND, IGO, ISTOP
COMMON /BLK5/ $FLAG, $TAG, $DOVEC
COMMON /BLK6/ EPSVIB, EPSVEC, EPSAIT
COMMON /BLK9/ ITR1, ITR2
COMMON /BLK10/ ASHIFT, ASHFUM, ATSHF, BII, TOLUP, TOLDWN, TOLCLS
C *****
WRITE(6,9999)
DELSO=1.0
IF(EPSVIB.LT.(1.0-08)) CALL VECOUT(3,&6)
6 IF(ITR1.LE.3) CALL VECOUT(2,&5)
DO 4 IVB=1,IVIBES
KEY1=MODVIB(1,IVB)
N(1)=MODVIB(2,IVB)
$DOVEC=.FALSE.
IF(MODVIB(3,IVB).EQ.1) $DOVEC=.TRUE.
IF(KEY1.EQ.4) CALL RANBER(&4)
IF(KEY1.EQ.1.AND.KEY1.LE.3) CALL VECOUT(6,&1)
SHFTPT(1,IVB)=0.0
$DOVEC=.TRUE.
KEY1=1
CALL VECOUT(5,&1)
DO 3 J=1,KEY1
1 ASHIFT=-(SHFTPT(J,IVB)+SNGUL(TWOPI)*TEEO)**2)
WRITE(6,8888) ASHIFT,SHFTPT(J,IVB)
K=J
IF(ASHIFT.EQ.(0.0)) K=1
CALL START(K)
CALL PMATRIX
CALL ITRREAL(&2)
CALL ITCPLX(&3)
GO TO 3
2 CALL VECOUT(9,&3)
3 CONTINUE
4 CONTINUE
5 RETURN
C *****
8888 FORMAT('0',52X,'SHIFT POINT IS',E14.6/66X,'(,E14.6,' HZ,')'
9999 FORMAT('///51X,'EXECUTING IN SUBROUTINE VIBERS'//49X,'PERFORMING

```





```

1FREE VIBRATION ANALYSES'////)
END

```

```

SUBROUTINE PMATRIX
C***
C*** THIS SUBROUTINE CALLS THE SUBROUTINES HJ(K,MN), EFG(K,MN), ABC,
C*** AND PANDD(K,MN) TO SET UP THE P, P-BAR AND P-HAT MATRICES GIVEN
C*** BY EQUATIONS (30) OF REFERENCE (2).
C*** INTERNALLY, MATRICES DL, DG AND DF ARE SET UP FOR THE CALCULA-
C*** TION OF X(1) GIVEN BY EQUATION (31A) OF REF. (2), WHERE
C***
C*** X(1) = DL*SMALL-L(1) + DG*SMALL-G(1) + DF*SMALL-F(1)
C***
C*** THE SPECIAL P MATRIX FOR A SHELL WITH AN INITIAL POLE IS ALSO
C*** COMPUTED HERE.
C*** MATRICES ZF1M, ZF2M, ZF3M, ZF4M ARE SET UP FOR THE CALCULATION OF
C*** Z(K+1) GIVEN BY EQUATION (31B) OF REF. (2), WHERE
C***
C*** Z(K+1)=ZF1M*SMALL-L(K) + ZF2M*X(K) + ZF3M*X(K-1) + ZF4M*SMALL-F
C*** IF THE SHELL HAS A FINAL POLE, THE MATRICES CL0,CL1,CL2 ARE
C*** PREPARED FOR THE CALCULATION OF Z(K)
C***
C*** IMPLICIT LOGICAL*1 ($)
C***
C*** REAL JAY
C*** COMMON /IBL1/ MNMAX
C*** COMMON /IBL2/ N(10), MNINIT
C*** COMMON /IBL3/ NO,M1,M2,M3
C*** COMMON /IBL4/ KMAX,KL
C*** COMMON /IBL5/ IBCINL,IBCFNL
C*** COMMON /IBL1/ A(4,4),BEE(4,4),C(4,4)
C*** COMMON /IBL4/ P(4,4,200),ZFIM(4,4,10),ZF2M(4,4,10),ZF3M(4,4,10),
C*** ZF4M(4,4,10)
C***
C*** 1 COMMON /BL13/ OMEG1(4,4),CAPL1(4,4),OMEGL(4,4),CAPLL(4,4),
C*** UNIT(4,4)
C***
C*** COMMON /BL23/ JAY(4,4),H(4,4)
C*** COMMON /BL24/ DG(4,4,10),DL(4,4,10),DF(4,4,10)
C*** COMMON /BL25/ E(4,4),F(4,4),G(4,4)
C*** COMMON /BL104/ Z(4,220),ZDOT(4,220),X(4,200),ZO(4,220),Z2(4,220),
C*** Z3(4,220)
C***
C*** 1 COMMON /BLK1/ VMAS,NN2,SVIBES
C*** COMMON /BLK10/ ASHIFT,ASHFUN,ATSHF,BII,TOLUP,TOLDWN,TOLCLS
C*** DIMENSION PATA(4,4),PBTE(4,4),POTA(4,4),DLL(4,4),PTR(4,
C*** 14),DGG(4,4),ZF1(4,4),ZF2(4,4),ZFPO(4,4),ZFP1(4,4),ZFP2(4,4),IPIVO
C*** 2T(4),INDEXI(4,2),CL0(4,4),CL1(4,4),CL2(4,4),G1(4)
C*** EQUIVALENCE (CL0(1),ZFIM(1)),(CL1(1),ZF2M(1)),(CL2(1),ZF3M(1)),
C*** 1(ZFPO(1),PATA(1)),(ZFPO(1),PBTA(1)),(ZFPO(1),POTA(1))
C***
9240
9250
9260
9270
9280
9290
9300
9310
9320
9330
9340
9350
9360
9370
9380
9390
9400
9410
9420
9430
9440
9450
9460
9470
9480
9490
9500
9510
9520
9530
9540
9550
9560
9570
9580
9590
9600
9610
9620
9630
9640
9650
9660
9670

```





```

2.(ZF1(1),DLL(1)),(ZF2(1),PTR(1))
  ASHFUM = ASHFIFT
  ATSHF = ASHFUM
  IF(IBC(INL,LT,0) GO TO 10
  DO 1 MN=MNINIT,MNMAX
  CALL HJ(1,MN)
  CALL EFG(1,MN)
  CALL ABC
  CALL MATINV(C,4,G1,0,DETERM,IPIVOT,INDEX,4,ISCALE)
  DO 3 I=1,4
  DO 3 J=1,4
  SUMA=0.
  SUMB=0.
  SUMC=0.
  SUMJ=0.
  DO 4 L=1,4
  SUMJ=SUMJ+OMEGI(I,L)*JAY(L,J)
  SUMA=SUMA+C(I,L)*A(L,J)
  SUMB=SUMB+C(I,L)*BEE(L,J)
  SUMC=SUMC+OMEGI(I,L)*H(L,J)
  PATA(I,J)=SUMA+UNIT(I,J)
  PBTA(I,J)=SUMB
  PCJA(I,J)=SUMC
  PCJA(I,J)=SUMJ
  DO 5 I=1,4
  DO 5 J=1,4
  SUMCB=0.
  SUMCA=0.
  SUMCC=0.
  DO 6 L=1,4
  SUMCB=SUMCB+POTA(I,L)*PATA(L,J)
  SUMCA=SUMCA+POTA(I,L)*PBTA(L,J)
  SUMCC=SUMCC+POTA(I,L)*PCJA(L,J)
  DLL(I,J)=SUMCB+PJTA(I,J)+CAPLI(I,J)
  PTR(I,J)=SUMCA
  EGG(I,J)=SUMCC
  CALL MATINV(DLL,4,PTR,4,DETERM,IPIVOT,INDEX,4,ISCALE)
  DO 1 I=1,4
  DO 1 J=1,4
  SUMD=0.
  SUME=0.
  DO 7 L=1,4
  SUMD=SUMD+DLL(I,L)*DGG(L,J)
  SUME=SUME-DLL(I,L)*CMEGI(L,J)
  EG(I,J,MN)=SUMD
  IF($VIPES) GO TO 2
  CF(I,J,MN)=SUME

```



```

      2 1 P(I,J,MN)=CLL(I,J)
      10 DO 11 MN=MNINIT,MNMAX
          NN=IABS(N(MN))
          IJ=1+KMAX*(MN-1)
          DO 14 I=1,4
              X(I,IJ)=0.
              DO 14 J=1,4
                  P(I,J,IJ)=0.
                  IF(NN.GT.3) GO TO 11
                  IF(NN.GT.2) GO TO 90
                  IF(NN.GT.1) GO TO 12
                  IF(NN.GT.0) GO TO 13
                  P(3,3,IJ)=-1.
                  P(4,4,IJ)=-1.
                  NO=MN
                  GO TO 11
              12 P(4,4,IJ)=-1.
                  GO TO 11
              90 M3=MN
                  GO TO 11
              13 P(1,1,IJ)=-1.
                  P(2,1,IJ)=1.
                  IF(N(MN).LT.0) P(2,1,IJ)=-1.
                  MI=MN
                  CC=INT(INUE
              11 KLAST=KMAX
              20 IF(I8CFNL.LT.0) KLAST=KL
                  DO 23 K=2,KLAST
                      CC 23 MN=MNINIT,MNMAX
                      CALL EFG(K,MN)
                      CALL ABC
              23 CALL PANDD(K,MN)
                  IF(I8CFNL.LT.0) GO TO 30
                  DO 40 MN=MNINIT,MNMAX
                      IKL=MN*KMAX-1
                      JKL=KMAX*MN
                      CALL HJ(KMAX,MN)
                      DO 41 I=1,4
                          DO 41 J=1,4
                              SUMQ=0.
                              SUMJ=0.
                              DO 42 L=1,4
                                  SUMQ=SUMQ+OMEG(L,I,L)*H(L,J)

```

10160  
10170  
10180  
10190  
10200  
10210  
10220  
10230  
10240  
10250  
10260  
10270  
10280  
10290  
10300  
10310  
10320  
10330  
10340  
10350  
10360  
10370  
10380  
10390  
10400  
10410  
10420  
10430  
10440  
10450  
10460  
10470  
10480  
10490  
10500  
10510  
10520  
10530  
10540  
10550  
10560  
10570  
10580  
10590  
10600  
10610  
10620  
10630



```

10640
10650
10660
10670
10680
10690
10700
10710
10720
10730
10740
10750
10760
10770
10780
10790
10800
10810
10820
10830
10840
10850
10860
10870
10880
10890
10900
10910
10920
10930
10940
10950
10960
10970
10980
10990
11000
11010
11020
11030
11040
11050
11060
11070
11080
11090
11100
11110

SUMP=SUMP+P(I,L,IKL)*P(L,J,JKL)
SUMJ=SUMJ+OMEG(L,I,L)*JAY(L,J)
PATA(I,J)=SUMO
PBTA(I,J)=UNIT(I,J)-SUMP
PJTA(I,J)=SUMJ+CAPLL(I,J)
DO 43 I=1,4
DO 43 J=1,4
SUMOP=0.
SUMJP=0.
SUMOM=0.
DO 44 L=1,4
SUMOP=SUMOP+PATA(I,L)*PBTA(L,J)
SUMJP=SUMJP+PJTA(I,L)*P(L,J,JKL)
SUMOM=SUMOM-PATA(I,L)*P(L,J,IKL)
ZF1(I,J)=SUMOP-SUMJP
ZF2(I,J)=SUMOM-PJTA(I,J)
CALL MATINV(ZF1,4,ZF2,4,DETERM,IPIVOT,INDEX,4,ISCALE)
DO 45 I=1,4
DO 45 J=1,4
SZF3=0.
SZF4=0.
DO 46 L=1,4
SZF3=SZF3+ZF1(I,L)*PATA(L,J)
SZF4=SZF4-ZF1(I,L)*OMEG(L,L,J)
ZF3M(I,J,MN)=SZF3
ZF4M(I,J,MN)=SZF4
ZF1M(I,J,MN)=ZF1(I,J)
ZF2M(I,J,MN)=ZF2(I,J)
CONTINUE
RETURN
DO 31 MN=MNINI T, MNMAX
IKL=VN*KMAX-1
NN=IABS(I(MN))
IF(NN.GT.3) GO TO 31
IF(NN.GT.2) GO TO 30
IF(NN.GT.1) GO TO 33
IF(NN.GT.0) GO TO 34
NO=MN
DO 35 J=1,4
DO 35 I=1,4
CLO(I,J)=0.
ZFO(I,J)=0.
ZFO(1,1)=1.
ZFO(2,2)=1.
ZFO(3,1)=P(3,1,IKL)
ZFO(3,2)=P(3,2,IKL)
ZFO(3,3)=P(3,3,IKL)+1.
ZFO(3,4)=P(3,4,IKL)

```



11120  
11130  
11140  
11150  
11160  
11170  
11180  
11190  
11200  
11210  
11220  
11230  
11240  
11250  
11260  
11270  
11280  
11290  
11300  
11310  
11320  
11330  
11340  
11350  
11360  
11370  
11380  
11390  
11400  
11410  
11420  
11430  
11440  
11450  
11460  
11470  
11480  
11490  
11500  
11510  
11520  
11530  
11540  
11550

```

ZFP0(4,1)=P(4,1,IKL)
ZFP0(4,2)=P(4,2,IKL)
ZFP0(4,3)=P(4,3,IKL)
ZFP0(4,4)=P(4,4,IKL)+1.
CLO(4,3)=1.
CLO(4,4)=1.
CALL MATINV(ZFP0,4,CLO,4,DETERM,IPIVOT,INDEX,4,ISCALE)
GO TO 31
300 M3=MN
GO TO 31
34 M1=MN
DO 60 J=1,4
  I=1,4
  CL1(I,J)=0.
  ZFP1(I,J)=P(1,1,IKL)+1.
  ZFP1(I,1)=P(1,2,IKL)
  ZFP1(I,2)=P(1,3,IKL)
  ZFP1(I,3)=P(1,4,IKL)
  ZFP1(I,4)=P(1,4,IKL)
  ZFP1(2,1)=1.
  ZFP1(2,2)=-1.
  IF(N(MN).LT.0) ZFP1(2,2)=1.
  ZFP1(3,3)=1.
  ZFP1(4,4)=1.
  CL1(1,1)=1.
  CALL MATINV(ZFP1,4,CL1,4,DETERM,IPIVOT,INDEX,4,ISCALE)
GO TO 31
33 M2=MN
DO 70 J=1,4
  I=1,4
  CL2(I,J)=0.
  ZFP2(I,J)=P(4,1,IKL)
  ZFP2(I,1)=P(4,2,IKL)
  ZFP2(I,2)=P(4,3,IKL)
  ZFP2(I,3)=P(4,4,IKL)+1.
  CL2(4,4)=1.
  CALL MATINV(ZFP2,4,CL2,4,DETERM,IPIVOT,INDEX,4,ISCALE)
31 CONTINUE
  RETURN
  END

```









```

COMMON /BL29/ BX1(10),BT1(10),EXT1(10),BEL(10),BX2(10),BT2(10),
1 COMMON /BL30/ BX12(10),BE2(10),
COMMON /BL31/ EXX1(10),ET1(10),EX1(10),ET1(10),EXX2(10),
1 COMMON /BL32/ EXT2(10),EXT2(10),EX2(10),ET2(10)
COMMON /BL33/ DELSQ,EXT1(10)
COMMON /BL34/ TEEQ,DYNAMIC
COMMON /BL35/ DELOA
COMMON /BL36/ MASS(200)
COMMON /BL37/ Z(4,220),ZDOT(4,220),X(4,200),Z0(4,220),Z2(4,220),
1 COMMON /BL38/ Z3(4,220)
COMMON /BL39/ ELLS(4),CL1(4,4),CL2(4,4)
1 COMMON /BL40/ CL1(4,4),ZDQ(4)
COMMON /BL41/ TZMAX(4,10),ZDQ(4)
2, TZMAX(4,10),ZDQ(4)
EQUIVALENCE (CL1(1),ZF1M(1)),(CL1(1),ZF2M(1)),(CL2(1),ZF3M(1))
C*****
DO 201 I=1,4
CC 201 M=1,MNMAX
NJ=1+(M-1)*KMAX2
TZMAX(I,M)=ABS(Z(I,NJ))
DO 201 K=2,KMAX2
KM=K+(M-1)*KMAX2
AZTST=ABS(Z(I,KM))
IF(AZTST.GT.TZMAX(I,M)) TZMAX(I,M)=AZTST
201 CONTINUE
IF(ITRMAX.EC.1) GO TO 66
CC 1 M=1,MNMAXQ
I=1+(KMAX+2)*(M-1)
U1(M)=Z(1,I)
V1(M)=Z(2,I)
W1(M)=Z(3,I)
I1=I+1
U2(M)=Z(1,I1)
V2(M)=Z(2,I1)
W2(M)=Z(3,I1)
1 IF(18CINL.LT.0) GO TO 100
CALL PHIBET(1)
CC 2 M=1,MNMAX
BX1(M)=BX3(M)
PT1(M)=BT3(M)
BX11(M)=BX13(M)
PE1(M)=BE3(M)
CALL TEAETA(1)
CC 3 M=1,MNMAX
EXX1(M)=EXX3(M)
ET11(M)=ETT3(M)
ETX1(M)=ETX3(M)

```



```

EXT1(M)=EXT3(M)
EX1(M)=EX3(M)
ET1(M)=ET3(M)
3  CALL PHIBET(2)
02  DO 4 M=1,MNMAX
    SX2(M)=SX3(M)
    BX2(M)=BT3(M)
    BF2(M)=BE3(M)
4  CALL TEAETA(2)
    DO 5 M=1,MNMAX
    EXX2(M)=EXX3(M)
    ETX2(M)=ETX3(M)
    EXT2(M)=EXT3(M)
5  EX2(M)=EX3(M)
    ET2(M)=ET3(M)
    CALL PHIBET(3)
    CALL TEAETA(3)
66  CONTINUE
    IF(IBCINL.LT.O) GO TO 20
    CALL BOR(1,BI,DB,D,CD)
    GAM1=GAM(1)
    CALL TLOAD(1)
    DO 8 M=1,MNMAX
    IF(ITRMAX.EQ.1) GO TO 67
    FFS(1,M)=-TT(M)*ALOAD+USE*(BX1(M)+BE1(M)+NU*(BT1(M)+BE1(M))*BI
    FFS(2,M)=OSE*(BI*CI
    FFS(3,M)=LAN2*GAM1*DI*MT(M)*ALOAD-(EXX1(M)+ETX1(M))*SOE
    GO TO 8
67  FFS(1,M)=-TT(M)*ALCAD
    FFS(2,M)=0.
    FFS(3,M)=LAM2*GAM1*DI*MT(M)*ALOAD
8  FFS(4,M)=0.
    DO 9 I=1,4
    ELIS(I)=ALOAD*EL1(I)
9  CALL FORCE(1)
20  CALL FORCE(2)
    DO 10 K=3,KLL
    KP=K+1
    IF(ITRMAX.EQ.1) GO TO 10
    CALL UPDATE
    CALL PHIBET(KP)
    CALL TEAETA(KP)
10  CALL FORCE(K)
    IF(ITRMAX.NE.1) CALL UPDATE
    IF(IBCENL.LT.O) GO TO 120
    IF(ITRMAX.EQ.1) GO TO 11

```





13000  
13010  
13020  
13030  
13040  
13050  
13060  
13070  
13080  
13090  
13100  
13110  
13120  
13130  
13140  
13150  
13160  
13170  
13180  
13190  
13200  
13210  
13220  
13230  
13240  
13250  
13260  
13270  
13280  
13290  
13300  
13310  
13320  
13330  
13340  
13350  
13360  
13370  
13380  
13390  
13400  
13410  
13420  
13430  
13440  
13450  
13460  
13470

```

CALL PHIBET(KMAX)
CALL TEAETA(KMAX)
CALL FORCE(KL)
CALL FORCE(KMAX)
DO 12 I=1,4
  ELLS(I)=ALOAD*ELL(I)
CALL BC8(KMAX,BL,CB,D,DD)
GAML=GAM(KMAX)
FLS(4)=0.
CALL TLOAD(KMAX)
DO 14 N=1,MNMAX
  IF(M.GT.1) ELLS(1)=0.0
  IF(ITRMAX.EQ.1) GO TO 68
  FLS(1)=-TT(M)*ALOAD+USE*(BX3(M)+BE3(M)+NU*(BT3(M)+BE3(M))*BL
  FLS(2)=-TT(M)*ALOAD+USE*(BX3(M)+EX3(M)+ET3(M))
  FLS(3)=L*AM2*GAML*DI*MT(M)*ALOAD-(EXX3(M)+ETX3(M))*SOE
GO TO 69
68 FLS(1)=-TT(M)*ALCAD
  FLS(2)=0.
  FLS(3)=L*AM2*GAML*DI*MT(M)*ALOAD
69 CONTINUE
  IK=KL+KMAX*(M-1)
  L=M*KMAX*M
DO 14 I=1,4
  SUMZ=0.
DO 15 J=1,4
  THE FOLLOWING CARD CAUSES BOUNDARY CONS TO EXIST FOR MODE 'O' ONLY
  IF(M.NE.1) ELLS(J)=0
  15 SUMZ=SUMZ+ZFIM(I,J,M)*ELL(J)
  14 Z(I,L)=SUMZ
  LS=1
DO 16 M=1,MNMAX
  DO 16 L=LS,KMAX
    K=KMAX2-L
    KPX=K-1
    KZ=K+1
    IJ=KPX+(M-1)*KMAX
    JK=KZ+(M-1)*KMAX2
    KK=JK-1
    DO 17 I=1,4
      SUMZ=0.
DO 18 J=1,4
  SUMZ=SUMZ-P(I,J,IJ)*Z(J,JK)
  SUMZ=SUMZ+X(I,IJ)
18

```





13480  
13490  
13500  
13510  
13520  
13530  
13540  
13550  
13560  
13570  
13580  
13590  
13600  
13610  
13620  
13630  
13640  
13650  
13660  
13670  
13680  
13690  
13700  
13710  
13720  
13730  
13740  
13750  
13760  
13770  
13780  
13790  
13800  
13810  
13820  
13830  
13840  
13850  
13860  
13870  
13880  
13890  
13900  
13910  
13920  
13930  
13940  
13950

```

17 ASUMZ=ABS(SUMZ)
16 IF(ASUMZ.GT.1.E+15) ITR=ITRMAX
   IF(NCONV.NE.1.OR.ASUMZ.LT.1.E-05) GO TO 17
   DELZ=ABS(Z(I,KK))-SUMZ
   ZTEST=EPS*TZMAX(I,M)
   IF(DELZ.GT.ZTEST) NCONV=0
   Z(I,KK)=SUMZ
   CONTINUE
   IF(IBCINL.LT.0) GO TO 30
   DO 25 M=1,MNMAX
   CALL EFG(I,M)
   CALL ABC
   IJ=2+(M-1)*KMAX2
   IJ1=IJ+1
   IJ2=IJ-1
   DO 21 I=1.4
   SUMZ=0.
   CO 22 J=1.4
   SUMZ=SUMZ-A(I,J)*Z(J,IJ1)-BEE(I,J)*Z(J,IJ)
   Z(I,I)=SUMZ+GEES(I,M)
   CALL MATINV(C,4,ZT,1,DETERM,IPIVOT,INDEX,4,ISCALE)
   DO 23 I=1.4
   Z(I,IJ2)=ZT(I)
   CCNTINUE
   RETURN
   CALL INLPOL
   DO 101 V=1,MNMAXC
   U1(M)=U2(M)
   V1(M)=V2(M)
   W1(M)=W2(M)
   IJ=3+KMAX2*(M-1)
   U2(M)=Z(1,IJ)
   V2(M)=Z(2,IJ)
   W2(M)=Z(3,IJ)
   GO TO 102
101 IF(ITRMAX.NE.1) CALL FNLPOL
120 CALL FORCE(KL)
   IF(M2.EQ.0) GO TO 122
   L=KL+(M2-1)*KMAX
   LI=KMAX1+(M2-1)*KMAX2
   DO 130 I=1.4
   SUM=0.
   CC 131 J=1.4
   SUM=SUM+CL2(I,J)*X(J,L)
   ASUMZ=ABS(SUM)
   IF(NCONV.NE.1.OR.ASUMZ.LT.1.E-05) GO TO 130
   DELZ=ABS(Z(I,L1))-SUM
   ZTEST=EPS*TZMAX(I,M2)

```



13960  
13970  
13980  
13990  
14000  
14010  
14020  
14030  
14040  
14050  
14060  
14070  
14080  
14090  
14100  
14110  
14120  
14130  
14140  
14150  
14160  
14170  
14180  
14190  
14200  
14210  
14220  
14230  
14240  
14250  
14260

```

130 IF(DELZ.GT.ZTEST) NCONV=0
122 Z(I,LI)=SUM
    IF(MI.EQ.0) GO TO 123
    L=KL+(MI-1)*KMAX
    LI=KMAX1+(MI-1)*KMAX2
    DO 132 I=1,4
        SUM=0.
        DO 133 J=1,4
            SUM=SUM+CL1(I,J)*X(J,L)
            ASUMZ=ABS(SUM)
            IF(NCONV.NE.1 .OR. ASUMZ.LT.1.E-05) GO TO 132
            DELZ=ABS(Z(I,LI)-SUM)
            ZTEST=EPS*ZMAX(I,MI)
            IF(DELZ.GT.ZTEST) NCONV=0
            Z(I,LI)=SUM
            IF(MI.EQ.0) GO TO 124
            L=KL+(MI-1)*KMAX
            LI=KMAX1+(MI-1)*KMAX2
            DO 134 I=1,4
                SUM=0.
                DO 135 J=1,4
                    SUM=SUM+CL0(I,J)*X(J,L)
                    ASUMZ=ABS(SUM)
                    IF(NCONV.NE.1 .OR. ASUMZ.LT.1.E-06) GO TO 134
                    DELZ=ABS(Z(I,LI)-SUM)
                    ZTEST=EPS*ZMAX(I,MI)
                    IF(DELZ.GT.ZTEST) NCONV=0
                    Z(I,LI)=SUM
                LS=2
            GO TO 150
        END
135
134
124

```

14270  
14280  
14290  
14300  
14310  
14320  
14330  
14340  
14350  
14360  
14370  
14380  
14390  
14400  
14410

```

SUBROUTINE FORCE(K)
** THIS SUBROUTINE COMPUTES THE GEE VECTOR IN EQUATION (28) OF
** REF. 2, AND THE X VECTOR IN EQUATION (29A) OF REF. 2 FOR A GIVEN MER-
** AT STATION K. THE VECTOR 'GEES' IS THE NONLINEAR VALUE OF 'GEE' **
** AT STATION 1. **
** IMPLICIT LOGICAL*1($) **
** REAL NU,MT,LAM2,MAS,MAS **
** COMMON /IBL1/ MNMAX **
** COMMON /IBL2/ N(10),MNIN:T **
** COMMON /IBL4/ KMAX,KL **
** COMMON /IBL8/ LSTEP,ITR **
** COMMON /IBL12/ KMAX1,KMAX2,NCONV **
** COMMON /IBL13/ ITRMAX,LSMAX **

```



```

CCCOMMON /BL3/ PR(10),PX(10),PT(10) 14420
CCCOMMON /BL4/ P(4,4,200),ZF1N(4,4,10),ZF2M(4,4,10),ZF3M(4,4,10), 14430
1 ZF4M(4,4,10) 14440
CCCOMMON /BL5/ TT(10),MT(10),DT(10),DMT(10) 14450
CCCOMMON /BL6/ SOE,OSE,ALOAD 14460
CCCOMMON /BL7/ DI,SI 14470
CCCOMMON /BL8/ R(200),GAM(200),DMT(200) 14480
CCCOMMON /BL9/ FFS(4,10),ELIS(4),CEES(4,10) 14490
CCCOMMON /BL11/ CMXI(200),PHEE,TO,T2 14500
CCCOMMON /BL12/ TDL1,TDEL 14510
CCCOMMON /BL14/ LAM2,LSDI8,LSDI9 14520
CCCOMMON /BL15/ NU,UI(10),V1(10),W1(10),V2(10),W2(10),U3(10), 14530
1 V3(10),W3(10) 14540
CCCOMMON /BL17/ DEL 14550
CCCOMMON /BL24/ DG(4,4,10),DL(4,4,10),DF(4,4,10) 14560
CCCOMMON /BL27/ PX3(10),BT3(10),BT3(10),BE3(10) 14570
CCCOMMON /BL28/ EXX3(10),ETT3(10),ETX3(10),EXT3(10),ET3(10) 14580
CCCOMMON /BL29/ BXT(10),BT1(10),BXT1(10),BE1(10),BX2(10),BT2(10), 14590
1 BXT2(10),BE2(10) 14600
CCCOMMON /BL30/ EXX1(10),ETT1(10),ETX1(10),ET1(10),EXX2(10), 14610
1 ETT2(10),ETX2(10),EXT2(10),ET2(10) 14620
CCCOMMON /BL31/ CELSO,EXT1(10) 14630
CCCOMMON /BL34/ CELSO(4,4,200),DST(4,4,200) 14640
CCCOMMON /PL100/ TEEC,$DYNMC 14650
CCCOMMON /BL101/ DELSD 14660
CCCOMMON /BL102/ DELLOAD 14670
CCCOMMON /BL103/ MASS(200) 14680
CCCOMMON /BL104/ Z(4,220),ZDCT(4,220),X(4,200),ZO(4,220),Z2(4,220), 14690
1 Z3(4,220) 14700
C** DIMENSION GEE(4) 14710
FDIFF(A,B,C)=(-1.5*A+2.0-0.5*C)/DEL 14720
RS=R(K) 14730
RR=1./RS 14740
GA=GAM(K) 14750
CX=CMXI(K) 14760
DT=DMT(K) 14770
CL2=DI+LAM2 14780
CALL BDB(K,BS,DBS,D,DD) 14790
CALL PLOAD(K) 14800
CALL TLOAD(K) 14810
YAS=MASS(K) 14820
DO 4 M=1,MNMAX 14830
IZ=K+1+(M-1)*KMAX 14840
IK=K+(M-1)*KMAX 14850
IK1=IK-1 14860
ENR=EN 14870
ENR=EN+RR 14880
ENR=EN+RR 14890

```













```

4 CONTINUE
RETURN
END

```

```

SUBROUTINE HJ(K,MN)
C THIS SUBROUTINE COMPUTES THE ELEMENTS OF THE H AND JAY
C MATRICES FOR BOTH RODDARIES OF THE SHELL
C
REAL L2,LAM2,LSDIN, LSD18,JAY,NU
COMMON /IBL2/ N(10),MNINIT
COMMON /IBL4/ KMAX,KL
COMMON /BL6/ R(200),GAM(200),OMT(200)
COMMON /BL11/ QMXI(200),PHEE,TO,t2
COMMON /BL14/ LAM2,LSD18,LSDIN
COMMON /BL15/ NU,U1(10),V1(10),W1(10),W2(10),U3(10),
1 V3(10),W3(10)
COMMON /BL17/ DEL
COMMON /BL20/ DEOMX(200)
COMMON /BL23/ JAY(4,4),H(4,4)
EQUIVALENCE(L2,LAM2)
CALL BDB(K,B,DB,D,OD)
YAH=1.
IF(K.EQ.1.OR.K.EQ.KMAX)YAH=2.
CI=(1.-NU)
GA=GAM(K)
OX=QMXI(K)
RA=R(K)
EN=N(MN)
ENQ=EN/RA
REG=0.
IF(YAH.EQ.2.) REG=1.
OT=OMT(K)
OXI=3.*OMXI(K)-OMT(K)
OTX=3.*OMT(K)-OMXI(K)
CL=DEL*L2*ENR
H(1,1)=R
H(1,2)=0.
H(1,3)=0.
H(1,4)=0.
H(2,1)=0.
H(2,2)=8.*D1/2.+L2*D*D1/8.*OTX*2*REG
H(2,3)=CL/2.*OTX*REG
H(2,4)=0.
H(3,1)=0.
H(3,2)=CL*OTX*YAH/4.

```

```

15860
15870
15880
15890
15900
15910
15920
15930
15940
15950
15960
15970
15980
15990
16000
16010
16020
16030
16040
16050
16060
16070
16080
16090
16100
16110
16120
16130
16140
16150
16160
16170
16180
16190
16200
16210
16220
16230
16240
16250
16260
16270
16280
16290
16300
16310

```



16320  
16330  
16340  
16350  
16360  
16370  
16380  
16390  
16400  
16410  
16420  
16430  
16440  
16450  
16460  
16470  
16480  
16490  
16500  
16510  
16520  
16530  
16540  
16550  
16560  
16570  
16580  
16590  
16600

```
ENR2=ENR**2
H(3,3)=L2*D*D1*(YAH*ENR2+(1.+NU)*GA**2)
GA2=GA**2
H(3,4)=L2
H(4,1)=0.
H(4,2)=0.
H(4,3)=-1.
H(4,4)=0.
JAY(1,1)=NU*GA*B
JAY(1,2)=NU*B*ENR
JAY(1,3)=B*(OX+NU*OT)
JAY(1,4)=0.
JAY(2,1)=-B*D1*ENR/2.-DL/8.*CXT*OTX*REG
JAY(2,2)=-GA*H(2,2)
JAY(2,3)=-GA*H(2,3)
JAY(2,4)=0.
JAY(3,1)=-L2*D*D1*((1.+NU)*GA2*OX+ENR2/4.*OXT*YAH)
JAY(3,2)=-GA*DL/2.*(2.*OT*(1.+NU)+OTX/2.*YAH)
JAY(3,3)=-L2*D*D1*((1.+NU+YAH)*GA*ENR2
JAY(3,4)=L2*D1*GA
JAY(4,1)=OX
JAY(4,2)=0.
JAY(4,3)=0.
JAY(4,4)=0.
DO 1 I=1,4
DO 1 J=1,4
1 H(I,J)=H(I,J)/2./DEL
END
```

16610  
16620  
16630  
16640  
16650  
16660  
16670  
16680  
16690  
16700  
16710  
16720  
16730  
16740  
16750  
16760  
16770

```
SUBROUTINE EFG(K,MN)
C THIS SUBROUTINE PREPARES THE ELEMENTS OF THE E, F, AND G
C FOR EACH MERIDIAN STATION K AND FOR EACH FOURIER MODE, MN.
C IMPLICIT LOGICAL*1 (F)
REAL NU,N,
LAW2,LSQIN,MASS,MAS
COMMON /BL2/NU(10),MASS,MAS
COMMON /BL8/ R(200),G(200),DMT(200)
COMMON /BL11/ CMXI(200),PHEE,TO,T2
COMMON /BL14/ LAW2,LSQIN,LSQIN
COMMON /BL15/ NU,U(10),V1(10),V2(10),W1(10),W2(10),U3(10),
1 V3(10),V4(10)
COMMON /BL20/ DECMX(200)
COMMON /BL25/ E(4,4),F(4,4),G(4,4)
COMMON /BL100/ TEEQ,AN,ANC
COMMON /BL101/ DELSD
```



```

COMMON /BL102/ DELOAD
COMMON /BL103/ MASS(200)
COMMON /BLK1/ VMAS,NN2,$VIBES
C*****
N=NN(MN)
MAS=MASS(K)
VMAS = MAS
IF($VIBES) MAS = 0.0
CALL BDG(K,B,DB,D,CD)
E(1,1)=B
E(1,2)=0.
E(1,3)=0.
E(1,4)=0.
E(2,1)=0.
C1=(1.-NU)
RA=R(K)
GA=GAM(K)
CX=CMXI(K)
CT=ONT(K)
LFX=DEOMX(K)
REX=(3.*CT-OX)
GA2=GA**2
RXE=(3.*OX-OT)
CTX=OT*OX
CNLR=LAM2*D*ND1/(2.*RA)
DNLR=DNLR*DD/D
E(2,2)=B*CI/2.+LAM2*D*CI*REX**2/8.
E(2,3)=CNLR*REX
E(2,4)=0.
E(3,1)=0.
E(3,2)=E(2,3)
RAN=(N/RA)**2
E(3,3)=LAM2*D*DI*(2.*RAN+(1.+NU)*GA2)
E(3,4)=LAM2
E(4,1)=0.
E(4,2)=0.
E(4,3)=0.
E(4,4)=0.
F(1,1)=GA*B+DB
F(1,2)=(1.+NU)*B*(N/RA)+DNLR*REX*RXE/4.
F(1,3)=B*(OX+NU*CT)+LAM2*D*DI*((1.+NU)*GA2*CX+RAN*RXE/2.)
F(1,4)=LAM2*OX
F(2,1)=-F(1,2)
F(2,2)=(DI/2.)*(GA*B+DB)-(LAM2*D*DI*REX/8.)*(2.*DEX-GA*(5.*OX
1-3.*OT))+LAM2*DD*CI*REX**2/8.
F(2,3)=DNLR*(2.*(1.+NU)*GA*OT-DEX+3.*GA*(CX-OT))+DDNLR*REX
F(2,4)=0.
F(3,1)=-F(1,3)

```

```

16780
16790
16800
16810
16820
16830
16840
16850
16860
16870
16880
16890
16900
16910
16920
16930
16940
16950
16960
16970
16980
16990
17000
17010
17020
17030
17040
17050
17060
17070
17080
17090
17100
17110
17120
17130
17140
17150
17160
17170
17180
17190
17200
17210
17220
17230
17240
17250

```





17260  
17270  
17280  
17290  
17300  
17310  
17320  
17330  
17340  
17350  
17360  
17370  
17380  
17390  
17400  
17410  
17420  
17430  
17440  
17450  
17460  
17470  
17480  
17490  
17500  
17510  
17520  
17530  
17540  
17550  
17560  
17570  
17580  
17590  
17600  
17610  
17620  
17630  
17640  
17650  
17660

17670  
17680  
17690  
17700  
17710

```

F(3,2)=DNLR*(3.*GA*CX-GA*OT*(5.+2.*NU)-DEX)+DDNLR*REX
F(3,3)=-LAM2*D*D1*((1.+NU)*(2.*GA*CX*OT+GA**3)+2.*GA*RX)
1+LAM2*D*D1*((1.+NU)*GA2+2.*RX)
F(3,4)=LAM2*GA*(2.-NU)
F(4,1)=D*QX
F(4,2)=0
F(4,3)=-D*NU*GA
F(4,4)=0
G(1,1)=NU*DB*GA-NU*B*OTX-B*GA2-D1*B*RX/2.-LAM2*D*D1*((1.+NU)*GA2*
1CX**2+RX**2*RX/8.)
2-2.*MAS/DELS
G(1,2)=NU*(DEX)
1+(1.+NU)*CX
G(1,3)=B*(DEX+GA*(OX-OT))+DB*(OX+NU*OT)-LAM2*D*D1*GA*RX*(RXE/2.+
11.+NU)*CX
G(1,4)=LAM2*D*D1*GA*CX
G(2,1)=-B*GA*N*(3.-NU)/(2.*RA)-D1*N*DB/(2.*RA)+DNLR*2.*(-1.*(1.+
1NU)*GA*OTX+GA/8.*(6.*OTX-7.*OX**2-3.*OT**2)-DEX/4.*(5.*OT-3.*OX))
2-DDNLR/4.*REX*RXE
G(2,2)=-GA*F(2,2)+D1/2.*B*OTX-B*RX-LAM2*D*D1*((1.+NU)*OT**2*RX
1-OTX/8.*REX**2)
2-2.*MAS/DELS
G(2,3)=-B*N*(OT+NU*CX)/PA+DNLR*(GA*DEX-2.*GA2*OX-2.*(1.+NU)*OT
1*RX+REX*(GA2+OTX))-DDNLR*REX*GA
G(3,1)=-B*GA*(OT+NU*CX)+LAM2*D*D1*(GA*(1.+NU)*(-GA*DEX+GA2*OX
1-OTX*RX+2.*CX*OX)+RX/2.*(GA*OX-GA*OT-3.*DEX))
2-LAM2*D*D1*((1.+NU)*GA2*OX+RX/2.*RXE)
G(3,2)=-B*N*(OT+NU*CX)/PA+DNLR*(2.*(1.+NU)*OTX*OT-GA2*OX+2.*GA2
1*OT-OT*RX)+GA*DEX+3.*GA2*(OT-OTX)+OTX*REX)-DDNLR*(2.*(1.+NU)*GA
2*OT+GA*REX)
G(3,3)=-B*(OX**2+2.*NU*OTX+OT**2)+LAM2*D*D1*RX*((1.+NU)*(OTX-RX
1+2.*GA2)+2.*(GA2+OTX))-LAM2*D*D1*RX*(3.+NU)*GA
2-2.*MAS/DELS
G(3,4)=-LAM2*(D1*OTX+NU*RX)
G(4,1)=D*(DEX+NU*GA*CX)
G(4,2)=D*(DEX+NU*OT/RA
G(4,3)=D*NU*RX
G(4,4)=-1.
RETURN
END

```

SUBROUTINE ABC  
C  
C THIS SUBROUTINE COMPUTES THE ELEMENTS OF THE A, BEE, AND C  
C MATRICES.  
C



17720  
17730  
17740  
17750  
17760  
17770  
17780  
17790  
17800  
17810  
17820  
17830  
17840  
17850  
17860  
17870  
17880  
17890  
17900  
17910  
17920  
17930  
17940  
17950  
17960  
17970  
17980  
17990

```

IMPLICIT LOGICAL*1 ($),BEE(4,4),C(4,4)
COMMON /BL1/ A(4,4),TDLI,TDEL
COMMON /BL12/ DEL
COMMON /BL25/ E(4,4),F(4,4),G(4,4)
COMMON /BLK1/ VMASS,NN2,$VIBES
COMMON /BLK10/ ASHIFT,ASHFUM,ATSHF,BII,TOLUP,TCLDWN,TOLCLS
COMMON /BLK11/ CIAG(4)
DIMENSION TEMP(48)
EQUIVALENCE (TEMP(1),A(1,1))
C2=2./DEL
DO 1 I=1,4
DO 1 J=1,4
  CEIJ=D2*E(I,J)
  FIJ=F(I,J)
  BEE(I,J)=-2.*DEIJ+TDEL*C(I,J)
  C(I,J)=DEIJ-FIJ
1  A(I,J)=DEIJ+FIJ
  IF(.NOT.$VIBES) GO TO 4
  VMASS=1./O(TDEL*VMASS)
  DO 2 I=1,48
  2  TEMP(I)=VMASS
  DO 3 I=1,4
  3  BEE(I,I)=BEE(I,I)-A*HFUM*C(I,I)
  CONTINUE
  4  END

```

18000  
18010  
18020  
18030  
18040  
18050  
18060  
18070  
18080  
18090  
18100  
18110  
18120  
18130  
18140  
18150  
18160  
18170

```

SUBROUTINE PANDD(K,MN)
**
** THIS SUBROUTINE COMPUTES THE ELEMENTS OF THE P, P-BAR, AND
** P-HAT MATRICES FOR EACH MEDIAN STATION K AND FOURIER MODE MN
** THESE MATRICES ARE COMPUTED AND SAVED BECAUSE THEY DO NOT
** CHANGE DURING EITHER THE ITERATION PROCEDURE OR THE LOAD INCRE-
** MENT PROCEDURE - AS THEY ARE A FUNCTION OF THE SHELL'S INITIAL
** GEOMETRY AND STIFFNESS.
**
COMMON /IBL4/ KMAX,KL
COMMON /BL1/ A(4,4),BEE(4,4),C(4,4)
COMMON /BL4/ P(4,4,200),ZF1M(4,4,10),ZF2M(4,4,10),ZF3M(4,4,10),
1  ZF4M(4,4,10)
COMMON /BL34/ DEE(4,4,200),DST(4,4,200)
DIMENSION TM(4,4),IPIVE(4),INDEX(4,2),X2(4)
**
IKL=K+KMAX*(MN-1)
KL=IKL-1

```



```

2 1 1 I=1,4
    1 J=1,4
    SUM=0.
    DO 2 L=1,4
        SUM=SUM+C(I,L)*P(L,J,KLI)
    TM(I,J)=BEE(I,J)-SUM
    1 CALL MATINV(TM,4,X2,0,DETERM,IPIVOT,INDEX,4,ISCALE)
    DO 5 I=1,4
        DO 5 J=1,4
            SUMA=0.
            SUMC=0.
            DO 6 L=1,4
                SUMA=SUMA+TM(I,L)*A(L,J)
                SUMC=SUMC+TM(I,L)*C(L,J)
            6 P(I,J,IKL)=SUMA
            DEE(I,J,IKL)=TM(I,J)
            DST(I,J,IKL)=SUMC
        5 RETURN
    END

```

```

18180
18190
18200
18210
18220
18230
18240
18250
18260
18270
18280
18290
18300
18310
18320
18330
18340
18350
18360

```











18830  
18840  
18850  
18860  
18870  
18880  
18890  
18900  
18910  
18920  
18930  
18940  
18950  
18960  
18970  
18980  
18990  
19000  
19010  
19020  
19030  
19040  
19050  
19060  
19070  
19080  
19090  
19100  
19110  
19120  
19130  
19140  
19150  
19160  
19170  
19180  
19190  
19200  
19210  
19220  
19230  
19240  
19250  
19260  
19270  
19280  
19290  
19300

```

DC 1 MM=NNS,MNMAXO
NM=N(N(MM))
NTEST=IABS(NMN-NMM)
DO 2 MMFT=1,MNMAX
IF(NTEST.EQ.N(MMFT)) GO TO 10
CONTINUE
2 IF(ICORFL.EQ.1) GO TO 1
NMMAX=MNMAX+1
N(MMFT)=NTEST
MMFT=MNMAX
IF(MNMAX.EQ.MAXM) ICORFL=1
10 IF(NMN-NMM) 11,1,12
11 LOCD=MAXD(MMFT)+1
MAXD(MMFT)=LOCD
12 LOCD,MMFT)=MM
JD(LOCD,MMFT)=MN
GO TO 1
LOCD=MAXD(MMFT)+1
MAXD(MMFT)=LOCD
11 LOCD,MMFT)=MN
JD(LOCD,MMFT)=MM
CONTINUE
1 DO 301 MN=1,MNMAXO
NNS=MN
N(MN)=N
IF(MNINIT.GT.MN) NNS=MNINIT
DO 301 MM=NNS,MNMAXO
NM=N(N(MM))
NTEST=MN+NMM
DO 302 MMFT=1,MNMAX
IF(NTEST.EQ.N(MMFT)) GO TO 310
CONTINUE
302 IF(ICORFL.EQ.1) GO TO 301
IF(MNMAX.GE.MAXM) GO TO 301
NMMAX=MNMAX+1
N(MMFT)=NTEST
MMFT=MNMAX
IF(MNMAX.GE.MAXM) ICORFL=1
310 IF(NMN.EQ.NMM) GO TO 360
LOCS=MAXS(MMFT)+1
MAXS(MMFT)=LOCS
IS(LOCS,MMFT)=MN
JS(LOCS,MMFT)=MM
GO TO 301
360 IF(NN.EQ.O) GO TO 301
MAXSY(MMFT)=1
IJS(MMFT)=MN
CONTINUE
301

```



19310  
19320  
19330  
19340  
19350

```

MNINIT=MNMAXO+1
IF(ICORFL.GT.0) IPASS=IPASS+1
IF(IPASS.LT.2.AND.MNINIT.LE.MNMAX) CALL PMATRIX
RETURN
END

```

19360  
19370  
19380  
19390  
19400  
19410  
19420  
19430  
19440  
19450  
19460  
19470  
19480  
19490  
19500  
19510  
19520  
19530  
19540  
19550  
19560  
19570  
19580  
19590  
19600  
19610  
19620  
19630  
19640  
19650  
19660  
19670  
19680  
19690  
19700  
19710  
19720  
19730  
19740  
19750  
19760

```

SUBROUTINE POLE(K)
C THIS SUBROUTINE PRINTS THE SOLUTION AT AN INITIAL AND A FINAL
C POLE.
C
C IMPLICIT LOGICAL*1 ($)
C REAL NU,MT,MX,MTH,MXT,MTS,KX,KT,KXT,LAM,LAM2,MASS
C COMMON /BL2/ N(13),MNINIT
C COMMON /BL3/ MO,M1,M2,M3
C COMMON /BL4/ KMAX,KL
C COMMON /BL5/ IBCFNL,IBCFNL
C COMMON /BL7/ MNMAXC,MAYC(10),MAXS(10),MAXSY(10),IS(10,10),
1 JS(10,10),ID(10,10),JD(10,10),IJS(10)
C COMMON /BL8/ LSTEP,ITR
C COMMON /BL10/ IFREC,NTHMAX
C COMMON /BL12/ KMAX1,KMAX2,NCONV
C COMMON /BL4/ P(4,4,200),ZF1N(4,4,10),ZF2M(4,4,10),ZF3M(4,4,10),
1 ZF4M(4,4,10)
C COMMON /BL5/ MT(10),DT(10),DMT(10)
C COMMON /BL6/ SCE,ALCAD
C COMMON /BL7/ OI,SI
C COMMON /BL8/ P(200),GAN(200),OMT(200)
C COMMON /BL10/ PHIX(10),PHIT(10),PHI(10)
C COMMON /BL11/ QMXI(200),PHEE,T0,T2
C COMMON /BL12/ TDLI,TDLI
C COMMON /BL14/ LAM2,LSD18,LSD1N
C COMMON /BL15/ NU,U(10),V(10),W1(10),W2(10),W3(10),U3(10),
1 V3(10),W3(10)
C COMMON /BL17/ DEL
C COMMON /BL19/ TH(36)
C COMMON /BL20/ DECMX(200)
C COMMON /BL27/ EX3(10),ST3(10),BXT3(10),BE3(10)
C COMMON /BL31/ CELSQ,EXT1(10)
C COMMON /BL32/ TKN,ELAST,CHAR,SIGO
C COMMON /BL100/ TEEQ,$DYNMC
C COMMON /BL101/ DELSQD
C COMMON /BL102/ DELCAD
C COMMON /BL103/ MASS(200)
C COMMON /BL104/ Z(4,220),ZDOT(4,220),X(4,200),ZO(4,220),Z2(4,220),
1 Z3(4,220)
C COMMON /BL110/ TX(10),TXI(10),MX(10),MTH(10),MXT(10),

```



```

1  COMMON /BULL11/ AEZ,ABZC,ABZN,ABZ3,DD2
C**      *
CALL BC8(K,BS,DB,DS,DC)
IF(K.EQ.KMAX) GO TO 301
DC 202 MN=1,MNMAXD
U1(MN)=U2(MN)
V1(MN)=V2(MN)
W1(MN)=W2(MN)
I3=3+(MN-1)*KMAX2
I2=I3-1
U2(MN)=Z(1,I3)
V2(MN)=Z(2,I3)
W2(MN)=Z(3,I3)
PHIX(MN)=0.0
PHIT(MN)=0.0
PHI(MN)=0.0
MX(MN)=Z(4,I2)*ABZ3
MTH(MN)=0.0
MXT(MN)=0.0
QS(MN)=0.0
TX(MN)=0.0
TTH(MN)=0.0
TXX(MN)=0.0
IF(M1.EQ.0) GO TO 203
CALL INLPOL
PHIX(M1)=PHEE*ABZO
PHIT(M1)=-PHEE*AEZO
QS(M1)=0.0
CONTINUE
IF(M2.EQ.0) GO TO 204
TX(M2)=T0*ABZ
TTH(M2)=T0*ABZ
MTH(M2)=MX(M2)
IF(M2.EQ.0) GO TO 205
TX(M2)=T2*ABZ
TTH(M2)=-T2*ABZ
MTH(M2)=-MX(M2)
MXT(M2)=MTH(M2)
GO TO 205
203 IF(M3.EQ.0) GO TO 206
I3=3+(M0-1)*KMAX2
I4=I3+1
CALL TLOAD(1)
TX(M0)=BS*SI*((2.*Z(1,I3))-5*Z(1,I4))/DEL+CMXI(1)*Z(3,I3-1)*ABZ
1 TTH(M0)=TX(M0)

```



```

206 MTH(MO)=MX(MO)
IF(M2.EQ.0) GO TC 205
I3=3+(M2-1)*KMAX2
I4=I3+1
TX(M2)=8S*01*(2.*Z(1,I3)-.5*Z(1,I4))/DEL
TX(M2) = TX(M2)*ABZ
TTH(M2) = -TX(M2)
TX(M2) = -TX(M2)
MTH(M2) = -MX(M2)
MX(M2) = -MX(M2)
RETURN
CONTINUE
DO 302 MN=1,MNMAXO
U1(MN)=U2(MN)
V1(MN)=V2(MN)
W1(MN)=W2(MN)
PHIX(MN)=0.
PHIT(MN)=0.
PHI(MN)=0.
IK=KMAX1+(MN-1)*KMAX2
MX(MN)=Z(4,IK)*ABZ3
MTH(MN)=0.
MX(MN)=0.
QS(MN)=0.
TX(MN)=0.
TTH(MN)=0.
TIT(MN)=0.
IF(M1.EQ.0) GO TO 303
CALL FNLPOL
PHIX(M1)=PHEE*ABZO
PHIT(M1)= PHEE*ABZO
QS(M1)=0.
CONTINUE
IF(MO.EQ.0) GO TO 304
TX(MO) = TO*ABZ
TTH(MO) = T)*ABZ
MTH(MO) = MX(MO)
IF(M2.EQ.0) GO TO 305
TX(M2) = T2*ABZ
TTH(M2) = -T2*ABZ
TX(M2) = -T2*ABZ
MTH(M2) = -MX(M2)
MX(M2) = MX(M2)
GO TO 305
303 IF(MO.EQ.0) GO TO 306
IKM=KMAX+(MO-1)*KMAX2
IMI=IKM-1
CALL TLOAD(KMAX)

```

20250  
20260  
20270  
20280  
20290  
20300  
20310  
20320  
20330  
20340  
20350  
20360  
20370  
20380  
20390  
20400  
20410  
20420  
20430  
20440  
20450  
20460  
20470  
20480  
20490  
20500  
20510  
20520  
20530  
20540  
20550  
20560  
20570  
20580  
20590  
20600  
20610  
20620  
20630  
20640  
20650  
20660  
20670  
20680  
20690  
20700  
20710  
20720





20730  
20740  
20750  
20760  
20770  
20780  
20790  
20800  
20810  
20820  
20830  
20840  
20850  
20860  
20870

```

TX(M0)=BS*S1*((-2.*Z(1,IKM)+.5*Z(1,IM1))/DEL+OMXI(KMAX)*Z(3,IKM+1)
) *ABZ-TT(M0)*ABZ*ALOAD
1 TTH(M0)=TX(M0)
MTH(M0)=MX(M0)
IF(M2.EQ.0) GO TO 305
306 IKM=KMAX+(M2-1)*KMAX2
IM1=IKM-1
TX(M2)=BS*DI*((-2.*Z(1,IKM)+.5*Z(1,IM1))/DEL
) *ABZ
TTH(M2)=TX(M2)
MTH(M2)=MX(M2)
MXT(M2)=MX(M2)
305 RETURN
END

```

20880  
20890  
20900  
20910  
20920  
20930  
20940  
20950  
20960  
20970  
20980  
20990  
21000  
21010  
21020  
21030  
21040  
21050  
21060  
21070  
21080  
21090  
21100  
21110  
21120  
21130  
21140  
21150  
21160  
21170  
21180

```

SUBROUTINE INLPOL
C THIS SUBROUTINE COMPUTES THE NON-LINEAR TERMS BETA-S,
C -SUB THETA, -SUB S-THETA, AETA-SUB S-S AND -SUB THETA-S AT AN
C INITIAL PCL.
C
C COMMON /IBL1/ MNMAX
C COMMON /IBL3/ M0,M1,M2,M3
C COMMON /IBL12/ KMAX1,KMAX2,NCONV
C COMMON /IBL13/ ITRMAX,LSMAX
C COMMON /BL5/ TT(10),ENT(10),DT(10),DMT(10)
C COMMON /BL6/ SOE,OSE,ALOAD
C COMMON /BL7/ DI,SI
C COMMON /BL11/ OMXI(200),PREE,TO,T2
C COMMON /BL17/ DEL
C COMMON /BL29/ BXT1(10),BXT1(10),BE1(10),BX2(10),BT2(10),
1 BXT2(10),SET(10)
1 EXX1(10),ETX1(10),ETX1(10),EX1(10),ET1(10),EXX2(10),
ETT2(10),ETX2(10),EX2(10),ET2(10)
1 CCMMON /BL30/ EXX1(10),EXX1(10),EXX1(10),EXX1(10),EXX1(10),
CCMMON /BL31/ DELSO,EX1(10)
1 CCMMON /BL104/ Z(4,220),Z(4,220),Z(4,220),Z(4,220),
Z(4,220)
C
C DC 1 MN=1, MNMAX
BXT1(MN)=0.
BT1(MN)=0.
BXT1(MN)=0.
BE1(MN)=0.
EX1(MN)=0.
ET1(MN)=0.
ETX1(MN)=0.

```



```

1 EXXI(MN)=0.
  IF(M1.EQ.0) RETURN
  I2=2+(M1-1)*KMAX2
  I3=I2+1
  I4=I3+1
  PHEE=(1.5*Z(3,I2)-2.*Z(3,I3)+.5*Z(3,I4))/DEL+OMXI(1)*Z(1,I2)
  BET=.5-PHEE**2
  IF(1+TRMAX.EQ.1) BET=0.
  T2=0.
  IF(M2.EQ.0) GO TO 2
  CALL BC9(1,B,DB,D,DD)
  I2=2+(M2-1)*KMAX2
  I3=I2+1
  I4=I3+1
  T2=B*DI*((-1.5*Z(1,I2)+2.*Z(1,I3)-.5*Z(1,I4))/DEL+.5*SOE*BET)
  Q1=.5*PHEE**T2
  BXT1(M2)=-BET
  BXT1(M2)=-BET
  ETT1(M1)=Q1
  IF(M3.EQ.0) GO TO 2
  EXXI(M3)=Q1
  ETT1(M3)=-Q1
  T1=0.
  IF(M0.EQ.0) GO TO 3
  BXT1(M0)=BET
  BXT1(M0)=BET
  CALL BC8(1,B,DB,D,DD)
  CALL TLOAD(1)
  I2=2+(M0-1)*KMAX2
  I3=I2+1
  I4=I3+1
  T0=B*SI*((-1.5*Z(1,I2)+2.*Z(1,I3)-.5*Z(1,I4))/DEL+OMXI(1)*Z(3,I2)
  1+.5*SOE*BET)-TT(M0)*ALQAD
  3 EXXI(M1)=PHEE*(1+.5-T2)
  RETURN
END

```

```

SUBROUTINE ENLPOL
C *****
C THIS SUBROUTINE COMPUTES THE NON-LINEAR TERMS BETA-SUB S, AT A
C -SUB THETA, -SUB S-THETA, AETA-SUB S-S, AND -SUB THETA-S, AT A
C FINAL POLE.
C *****
COMMON /IBL1/ MNMAX
COMMON /IBL3/ M0,M1,M2,M3
COMMON /IBL4/ KMAX,KL
21560
21570
21580
21590
21600
21610
21620
21630
21640

```



```

COMMON /IBL12/ KMAX1,KMAX2,NCCNV
COMMON /IBL13/ ITRMAX,LSIAX
COMMON /BL5/ TT(10),EMT(10),DT(10),DMT(10)
COMMON /BL6/ SOE,CSE,ALC10
COMMON /BL7/ CI,SI
COMMON /BL11/ OMXI(200),PHEE,T0,T2
COMMON /BL17/ DEL
COMMON /BL27/ BX3(10),BT3(10),BE3(10)
COMMON /BL28/ EXX3(10),ET3(10),EX3(10),ET3(10)
COMMON /BL104/ Z(4,220),Z0(4,200),X(4,200),Z2(4,220),
1 Z3(4,220)
C*****
DO 1 MN=1,MNMAX
BX3(MN)=0.
BT3(MN)=0.
EXX3(MN)=0.
ET3(MN)=0.
EXX3(MN)=0.
ET3(MN)=0.
EXX3(MN)=0.
ET3(MN)=0.
EXX3(MN)=0.
ET3(MN)=0.
CALL BOB(KMAX,B,DB,D,DB)
IF(M1.EQ.0) RETURN
KM=KMAX1+(M1-1)*KMAX2
KM1=KM-1
KM2=KM-2
PHEE=-(1.5*Z(3,KM)-2.*Z(3,KM1)+.5*Z(3,KM2))/DEL+OMXI(KMAX)*Z(1,KM)
BT=5*PHEE**2
IF(ITRMAX.EQ.1) BET=0.
T2=0.
IF(M2.EQ.0) GO TO 2
KM=KMAX1+(M2-1)*KMAX2
KM1=KM-1
KM2=KM-2
T2=B**2*(1.5*Z(1,KM)-2.*Z(1,KM1)+.5*Z(1,KM2))/DEL+.5*SOE*BET
Q1=.5*PHEE**T2
BX3(M2)=-BET
BT3(M2)=BET
BTX3(M1)=Q1
ETX3(M3)=Q1
FXX3(M3)=Q1
ETX3(M3)=-Q1
TJ=0.
IF(M0.EQ.0) GO TO 3
CALL TLOAD(KMAX)
KM=KMAX1+(M0-1)*KMAX2
KM1=KM-1
2

```





22130  
22140  
22150  
22160  
22170  
22180  
22190  
22200

```

KM2=KM-2
BX3(MO)=BET
BT3(MO)=BET
TC=B*SI*(1.5*SOE*BET)-TT(PJ)*ALOAD
1 Z(3,KM)+.5*Z(1,KM)-2.*Z(1,KM1)+.5*Z(1,KM2))/DEL+OMXI(KMAX)*
3 EXX3(M1)=PHEE*(TC+.5*T2)
RETURN
END

```

22210  
22220  
22230  
22240  
22250  
22260  
22270  
22280  
22290  
22300  
22310  
22320  
22330  
22340  
22350  
22360  
22370  
22380  
22390  
22400  
22410  
22420  
22430  
22440  
22450  
22460  
22470  
22480  
22490  
22500  
22510  
22520  
22530  
22540  
22550  
22560  
22570  
22580

```

C** SUBROUTINE PHIBET(K)
C**
C** THIS SUBROUTINE CALCULATES THE PHI'S AND CARRIES OUT THE
C** MULTIPLYING AND SUMMATION PROCEDURE FOR COMPUTING THE BETA
C** NON-LINEAR TERMS FOR A GIVEN PERIODICAL STATION K. THE ARRAYS
C** IS, JS, ID, JS, IJS, MAXS, MAXD, MAXSY ARE PREPARED IN SUB-
C** ROUTINE MODES AND USED HERE.
C**
C** COMMON /IBL1/ MNMAX
C** COMMON /IBL2/ N(10), MNINIT
C** COMMON /IBL4/ KMAX, KL
C** COMMON /IBL7/ J(10,10), MAXS(10), MAXSY(10), IS(10,10),
1 JS(10,10), ID(10,10), JD(10,10), IJS(10,10)
C** COMMON /IBL12/ KMAX1, KM, X2, NCONV
C** COMMON /IBL13/ ITRMAX, L, MAX
C** COMMON /IBL6/ SQ, QSE, AL, AD
C** COMMON /BL8/ R(200), GA(200), OMT(200)
C** COMMON /BL10/ PHIX(10), PHIT(10), PHI(10)
C** COMMON /BL11/ OMXI(200), PHEE, TO, T2
C** COMMON /BL12/ TDLI, TCEL
C** COMMON /BL15/ NU, UI(10), V1(10), V2(10), U2(10), W2(10), U3(10),
1 V3(10), W3(10)
C** COMMON /BL27/ BX3(10), BXT3(10), BE3(10)
C** COMMON /BL104/ Z(4,220), ZDCT(4,200), X(4,200), ZC(4,220),
1 Z3(4,220)
C**
C** OX=OMXI(K)
C** CT=CMT(K)
C** RRA=1./R(K)
C** GA=GAM(K)
C** KP2=K+2
C** DO 1 M=1, MNMAXC
C** EN=N(M)
C** IK=KP2+(M-1)*KMAX2
C** U3(M)=Z(1,IK)
C** V3(M)=Z(2,IK)
C** W3(M)=Z(3,IK)
C** PHIX(M)=-TDLI*(W3(M)-X1(Y))+OX*U2(M)

```





```

1 PHIT(M)=EN*W2(M)*RRA+V2(M)*QT
  IF(ITERMAX.EQ.1) RETURN
  DO 9 M=1,MNMAX
    SMO=0.
    SMT=0.
    SMR=0.
    SMF=0.
    IF(N(M).EQ.0) GO TO 20
    MAXL=MAXS(M)
    IF(MAXL.EQ.0) GO TO 2
    DO 3 L=1,MAXL
      I=IS(L,M)
      J=JS(L,M)
      SMO=SMO+PHIX(I)*PHIX(J)
      SMT=SMT-PHIT(I)*PHIT(J)
      SMR=SMR+PHIX(I)*PHIT(J)+PHIX(J)*PHIT(I)
      SMF=SMF-PHI(I)*PHI(J)
      MAXL=MAXD(M)
      IF(MAXL.EQ.0) GO TO 4
      DO 5 L=1,MAXL
        I=ID(L,M)
        J=JD(L,M)
        SMC=SMO+PHIX(I)*PHIX(J)
        SMT=SMT+PHIT(I)*PHIT(J)
        SMR=SMR+PHIX(I)*PHIT(J)+PHIX(J)*PHIT(I)
        SMF=SMF+PHI(I)*PHI(J)
        IF(MAXSY(M).EQ.0) GC TO 10
        I=IJS(M)
        J=JCS(M)
        SMC=SMO+PHIX(I)*2/2.
        SMT=SMT-PHIT(I)*2/2.
        SMR=(SMR+PHIX(I)*PHIT(I))
        SMF=SMF-PHI(I)*2/2.
        GO TO 10
      DO 21 L=1,MNMAXO
        SMC=SMO+PHIX(L)*2
        SMT=SMT+PHIT(L)*2
        SMR=SMR+PHIX(L)*2
        SMF=SMF+PHI(L)*2
        IF(M.GT.MNMAXO) GO TO 11
        SMO=SMO+PHIX(M)*2
        BX3(M)=SMO*.5
        BT3(M)=SMT*.5
        BXT3(M)=SMF*.5
        GO TO 5
      DO 10 L=1,5
        BX3(M)=SMO
        BT3(M)=SMT
        BXT3(M)=SMR*.5

```

2590  
 22600  
 22610  
 22620  
 22630  
 22640  
 22650  
 22660  
 22670  
 22680  
 22690  
 22700  
 22710  
 22720  
 22730  
 22740  
 22750  
 22760  
 22770  
 22780  
 22790  
 22800  
 22810  
 22820  
 22830  
 22840  
 22850  
 22860  
 22870  
 22880  
 22890  
 22900  
 22910  
 22920  
 22930  
 22940  
 22950  
 22960  
 22970  
 22980  
 22990  
 23000  
 23010  
 23020  
 23030  
 23040  
 23050  
 23060



23070  
23080  
23090  
23100

BE3(M)=SMF  
9 CONTINUE  
RETURN  
END

23110  
23120  
23130  
23140  
23150  
23160  
23170  
23180  
23190  
23200  
23210  
23220  
23230  
23240  
23250  
23260  
23270  
23280  
23290  
23300  
23310  
23320  
23330  
23340  
23350  
23360  
23370  
23380  
23390  
23400  
23410  
23420  
23430  
23440  
23450  
23460  
23470  
23480  
23490  
23500  
23510  
23520

```

SUBROUTINE TEAETA(K)
C THIS SUBROUTINE CALCULATES THE INPLANE FORCES AND CARRIES OUT
C THE MULTIPLYING AND SUMMATION PROCEDURE FOR COMPUTING THE AETA
C NON-LINEAR TERMS FOR A GIVEN PERIODICAL STATION K. THE ARRAYS
C IS, JS, ID, JS, IJS, MAXS, MAXC, MAXSY PREPARED IN SUBROUTINE
C MODES ARE USED HERE.
C
REAL NU, MT
COMMON /IBL1/ MNMAX
COMMON /IBL2/ N(10), MNMAXC, MAXC(10), MAXS(10), MAXSY(10), IS(10,10),
COMMON /IBL7/ JS(10,10), ID(10,10), JD(10,10), IJS(10,10),
1 LSTEP, ITR, CMAX
COMMON /IBL8/ LSTEPMAX, LCMAX
COMMON /IBL13/ IT(10), NI(10), DT(10), DMT(10)
COMMON /BL5/ SOE, OSE, ALCOA
COMMON /BL7/ SI, S1
COMMON /BL8/ R(200), CMT(200)
COMMON /BL10/ PHIX(10), PHIT(10), PHI(10)
COMMON /BL11/ QMXI(200), PHEE, T0, T2
COMMON /BL12/ TDLI, TDEL
COMMON /BL15/ NU, UI(10), V1(10), V2(10), U2(10), W2(10), U3(10),
1 V3(10), W3(10)
COMMON /BL27/ BX3(10), BXT3(10), BE3(10)
COMMON /BL28/ EXX3(10), ETX3(10), EX3(10), ET3(10)
DIMENSION TX(10), TTH(10), TTX(10)
C
PRA=1./R(K)
GA=GAM(K)
CX=CMXI(K)
CT=CMT(K)
CALL BCB(K, BS, DB, DS, DD)
DO 1 M=1, MNMAXG
EN=N(M)
CALL TLOAD(K)
TTS=TT(M)*ALOAD
EX=(U3(M)-UI(M))*TDLI+CX*J2(M)+OSE*(BX3(M)+BE3(M))
ET=EN*V2(M)*RRA+GA*U2(M)+OT*W2(M)+OSE*(BT3(M)+BE3(M))
EXT=.5*(TDLI*(V3(M)-V1(M))-EN*U2(M)*RRA-GA*V2(M))+OSE*BXT3(M)
TX(M)=BS*(CX+NU*ET)-TT
TTH(M)=BS*(ET+NU*EX)-TT

```



1	IXT(M)=BS*DI*EXT	23530
	DO 9 M=1,MNMAX	23540
	SMF=0.	23550
	SMV=0.	23560
	SME=0.	23570
	SMN=0.	23580
	SMT=0.	23590
	IF(N(M).EQ.0) GO TO 20	23600
	MAXL=MAX(XS(M))	23610
	IF(MAXL.EQ.0) GO TO 2	23620
	DO 3 L=1,MAXL	23630
	I=IS(L,M)	23640
	J=JS(L,M)	23650
	SMF=SMF+TX(I)*PHIX(J)+TX(J)*PHIX(I)	23660
	SMS=SMS+TTH(I)*PHIT(J)+TTH(J)*PHIT(I)	23670
	SMV=SMV-PHIT(I)*TXI(J)-PHIT(J)*TXI(I)	23680
	SME=SME+PHIX(I)*TXI(J)+PHIX(J)*TXI(I)	23690
	SMN=SMN+TX(I)*PHI(J)+TX(J)*PHI(I)	23700
	SMT=SMT+TTH(I)*PHI(J)+TTH(J)*PHI(I)	23710
	MAXL=MAX(XD(M))	23720
3	IF(MAXL.EQ.0) GO TO 4	23730
2	DO 5 L=1,MAXL	23740
	I=ID(L,M)	23750
	J=JD(L,M)	23760
	SMF=SMF+TX(I)*PHIX(J)+TX(J)*PHIX(I)	23770
	SMS=SMS-TTH(I)*PHIT(J)+TTH(J)*PHIT(I)	23780
	SMV=SMV+PHIT(I)*TXI(J)+PHIT(J)*TXI(I)	23790
	SME=SME-PHIX(I)*TXI(J)+PHIX(J)*TXI(I)	23800
	SMN=SMN-TX(I)*PHI(J)+TX(J)*PHI(I)	23810
	SMT=SMT-TTH(I)*PHI(J)+TTH(J)*PHI(I)	23820
5	IF(MAXSY(M).EQ.0) GO TO 10	23830
4	I=IJS(M)	23840
	SMF=SMF+TX(I)*PHIX(I)	23850
	SMS=SMS+TTH(I)*PHIT(I)	23860
	SMV=SMV-PHIT(I)*TXI(I)	23870
	SME=SMF+PHIX(I)*TXI(I)	23880
	SMN=SMN+TX(I)*PHI(I)	23890
	SMT=SMT+TTH(I)*PHI(I)	23900
	GO TO 10	23910
20	GO 21 L=1,MNMAXO	23920
	SMF=SMF+TX(L)*PHIX(L)	23930
21	SMV=SMV+PHIT(L)*TXI(L)	23940
	IF(M.GT.MNMAXO) GO TO 10	23950
	SMF=SMF+TX(M)*PHIX(M)	23960
10	EXX3(M)=SMF*.5	23970
	ETT3(M)=SMS*.5	23980
	ETX3(M)=SMV*.5	23990
		24000



```

EXT3(M)=SME*.5
EX3(M)=SMN*.5
ET3(M)=SMT*.5
CCNTINUE
9 DO 30 M=1,MNMAXO
  U1(M)=U2(M)
  V1(M)=V2(M)
  W1(M)=W2(M)
  U2(M)=U3(M)
  V2(M)=V3(M)
  W2(M)=W3(M)
30 RETURN
END

```

24010  
24020  
24030  
24040  
24050  
24060  
24070  
24080  
24090  
24100  
24110  
24120  
24130

```

SUBROUTINE UPDATE
C *** THIS SUBROUTINE UPDATES THE STORAGE LOCATIONS OF THE BETA'S AND
C *** ETAS. IT IS CALLED IN SUBROUTINE XANDZ AFTER A MERIDIAN
C *** STATION CHANGE. ***
C ***
COMMON /IRL1/ MNMAX
COMMON /BL27/ EXX3(10),ETX3(10),EX3(10),BT3(10)
COMMON /BL28/ EXX1(10),ETX1(10),BX2(10),BT2(10),
COMMON /BL29/ EXX2(10),ETX2(10),EX1(10),BT1(10)
1 COMMON /BL30/ EXX1(10),ETX1(10),EX2(10),EXX2(10),
1 ETX2(10),ETX2(10),EX2(10),ET2(10)
C *** CELSO, XT1(10) ***
COMMON /BL31/
C ***
CO 1 M=1,MNMAX
BX1(M)=BX2(M)
BT1(M)=BT2(M)
BX11(M)=BX12(M)
BX2(M)=BX2(M)
BT2(M)=BT3(M)
BX22(M)=BX23(M)
EXX1(M)=EXX2(M)
ETX1(M)=ETX2(M)
ETX11(M)=ETX2(M)
EXX1(M)=EXX2(M)
ETX1(M)=ETX2(M)
EXX2(M)=EXX3(M)
ETX2(M)=ETX3(M)
ETX22(M)=ETX3(M)

```

24140  
24150  
24160  
24170  
24180  
24190  
24200  
24210  
24220  
24230  
24240  
24250  
24260  
24270  
24280  
24290  
24300  
24310  
24320  
24330  
24340  
24350  
24360  
24370  
24380  
24390  
24400  
24410  
24420  
24430  
24440  
24450  
24460





```

EXT2(M)=EXT3(M)
EXT2(M)=EXT3(M)
1 RETURN
END

```

```

24470
24480
24490
24500
24510

```

```

SUBROUTINE OUTPUT(IMODE)
C *****
C THIS SUBROUTINE PREPARES THE PRINTOUT MATERIAL. EVERY IPRINT
C CONVERGED SOLUTION IS PRINTED. THE FOURTH COEFFICIENTS OF THE
C INPLANE FORCES, MERIDIONAL TRANT, NSVERSE FORCE, CIRCUMFERENTIAL
C BENDING MOMENT, TWISTING MOMENT AND ROTATIONS CAN BE COMPUTED
C AND PRINTED WITH THE SOLUTIONS FOR THE FOURIER COEFFICIENTS. THIS
C OF THE THREE-DIAL IS CONVERSION FROM DIMENSIONLESS FORM TO DIMEN-
C OUTPUT MATERIAL HERE. I, ETC. THIS SUBROUTINE PRINT ONLY AT STATIONS
C SIONAL FORM HERE. I, ETC. THIS SUBROUTINE PRINT ONLY AT STATIONS
C I, IFREQ+1, PROCESS RESULTS AND ROTATIONS. ALSO PERFORMS THE
C SUMMATIONS, DISPLACEMENTS AND PRESCRIPTIONS AT THE INPUT POSITIONS.
C AROUND THE CIRCUMFERENCE PREPARED HERE, IF REQ'D *****
C DATA TO BE PLOTTED IS PREPARED HERE, IF REQ'D *****
C *****
C IMPLICIT LOGICAL*1 (S)
C REAL NU,MT,MX,MTH,MXT,MTS,KX,KT,KXT,LAM,LAM2,MASS
C COMMON /IBL2/ N(10),M1,M2,M3
C COMMON /IBL3/ M1,M2,M3
C COMMON /IBL4/ KMAX,KL
C COMMON /IBL5/ IBCINL,IBCENL,IBCENL
C COMMON /IBL7/ MNMAXD,MAXD(10),ID(10,10),JD(10,10),IJS(10)
C *****
1 CCMCN /IBL8/ LSTEP,ITR
C CCMCN /IBL10/ IFREQ,NTHMAX
C CCMCN /IBL12/ KMAX1,KMAX2,NCONV
C CCMCN /IBL13/ ITRMAX,LSMAX
C CCMCN /IBL4/ P(4,4,200),ZF1M(4,4,10),ZF2M(4,4,10),ZF3M(4,4,10),
C ZF4M(4,4,10)
1 CCMCN /BL5/ ZF4M(4,4,10),DT(10),DMT(10)
C CCMCN /BL6/ SOE,DOSE,ALOAD
C CCMCN /BL7/ DI,SI
C CCMCN /BL8/ R(200),GAM(200),GMT(200)
C CCMCN /BL10/ PHIX(10),PHIT(10),PHI(10)
C CCMCN /BL11/ CMXI(200),PHEE,T0,T2
C CCMCN /BL12/ TDLI,TOEL
C CCMCN /BL14/ LAM2,LSCL8,LSDIN
C CCMCN /BL15/ NU,U1(10),V1(10),W1(10),V2(10),U2(10),W2(10),U3(10),
C V3(10),W3(10)
1 CCMCN /BL17/ DEL
C *****

```

```

24520
24530
24540
24550
24560
24570
24580
24590
24600
24610
24620
24630
24640
24650
24660
24670
24680
24690
24700
24710
24720
24730
24740
24750
24760
24770
24780
24790
24800
24810
24820
24830
24840
24850
24860
24870
24880
24890
24900
24910
24920

```







```

1 ICHCK2=IABS(IU)+IABS(IMSTH)+IABS(IPHIS)+IABS(IPHIT)
1 IF(NTHMAX.EQ.0) GO TO 991
IF($VIRE$) GO TO 991
DO 21 NTH=1,NTHMAX
DO 1 MN=1,MNMAXO
I1=1+(MN-1)*KMAX2
I2=I1+1
U1(MN)=Z(1,I1)
U2(MN)=Z(1,I2)
V1(MN)=Z(2,I1)
V2(MN)=Z(2,I2)
W1(MN)=Z(3,I1)
W2(MN)=Z(3,I2)
THET=TH(NTH)
WRITE(6,I16) THET
DO 121 K=1,KMAX
K1=K+1
CALL BD9(K,BS,DB,DS,DD)
IF(K.EQ.1.AND.IBCINL.LT.0) CALL POLE(K)
IF(K.EQ.1.AND.IBCINL.LT.0) GO TO 999
IF(K.EQ.KMAX.AND.IBCFNL.LT.0) CALL POLE(K)
IF(K.EQ.KMAX.AND.IBCFNL.LT.0) GO TO 999
CALL PHIBET(K)
DEX=DEOMX(K)
BRA=1./R(K)
OX=OMX1(K)
OT=OMT(K)
GA=GAM(K)
COXT=OX-OT
GDO=GAXDCXT
DO2D=DC2*DS
DO 3 MN=1,MNMAXO
ENR=EN(MN)
ENR=EN*RA
CALL TLOAD(K)
TTS=TT(MN)*ALOAD
EX=(U3(MN)-U1(MN))*TDLI+OX*W2(MN)+ENL*QSE*(BX3(MN)+BE3(MN))
ET=ENR*V2(MN)+GA*U2(MN)+OT*W2(MN)+ENL*QSE*(BT3(MN)+BE3(MN))
EXT=.5*((V3(MN)-V1(MN))*TDLI-ENR*U2(MN)-GA*V2(MN)+ENL*SOE*BXT3(MN))
1) KT=ENR*PHIT(MN)+GA*PHIX(MN)
KXT=.5*(ENR*(-PHIX(MN)-GA*W2(MN)+W3(MN))-W1(MN))*TDLI+GDO*V2(MN)
1) OT*(V3(MN)-V1(MN))*TDLI-GA*PHIT(MN)-DCXT*PHI(MN)
TX(MN)=BS*(EX+NU*ET)-TTS
TTH(MN)=BS*(ET+NU*EX)-TTS
TXT(MN)=BS*D1*EXT

```





```

MK1=K1+(MN-1)*KMAX2
MX(MN)=Z(4,MK1)
MTH(MN)=NU*MX(MN)+DD2D*KT-D1*NT(MN)*ALOAD
MXT(MN)=DS*D1*KXT
MK11=MK1+1
MKK1=MK1-1
1  QS(MN)=SIGO*TKN*LAM2*(GA*MX(MN)+(Z(4,MK11)-Z(4,MK1))*TDLI
    +ENR*MX(MN)-GA*MTH(MN))
    MX(MN)=MX(MN)*ABZ3
    MTH(MN)=MTH(MN)*ABZ3
    MXT(MN)=MXT(MN)*ABZ3
    TX(MN)=TX(MN)*ABZ
    TTH(MN)=TTH(MN)*ABZ
    TXT(MN)=TXT(MN)*ABZ
    PHIX(MN)=PHIX(MN)*ABZO
    PHIT(MN)=PHIT(MN)*ABZO
    PHI(MN)=PHI(MN)*ABZC
    U1(MN)=U2(MN)
    U2(MN)=U3(MN)
    V1(MN)=V2(MN)
    V2(MN)=V3(MN)
    W1(MN)=W2(MN)
    W2(MN)=W3(MN)
    FK=K-1
    IFREQ=IFREQ
    KTST=(K-1)/IFREQ
    FKST=KTST
    FKTEST=FK/IFREQ-FKTST
    IF(K.EC.1-GR.K.EC.KMAX) GO TO 999
    IF(FKTEST.NE.0.) GC TC 2
999 X(1,K)=0.
    X(2,K)=0.
    X(3,K)=0.
    X(4,K)=0.
    PTF(K)=0.
    PF(K)=0.
    AMX=0.
    AMTH=0.
    AMXTH=0.
    ANX=0.
    ANTH=0.
    ANXTH=0.
    AQS=0.
    DO 72 MN=1,MNMAXO
    EA=N(MN)
    EC=EN*THET
    SA=SIN(EC)
    CS=COS(EC)

```





26370  
26380  
26390  
26400  
26410  
26420  
26430  
26440  
26450  
26460  
26470  
26480  
26490  
26500  
26510  
26520  
26530  
26540  
26550  
26560  
26570  
26580  
26590  
26600  
26610  
26620  
26630  
26640  
26650  
26660  
26670  
26680  
26690  
26700  
26710  
26720  
26730  
26740  
26750  
26760  
26770  
26780  
26790  
26800  
26810  
26820  
26830  
26840

```

X(1,K)=X(1,K)+U1(MN)*CS*ABZN
X(2,K)=X(2,K)+V1(MN)*SN*ABZN
X(3,K)=X(3,K)+W1(MN)*CS*ABZN
X(4,K)=X(4,K)+PHI1(MN)*CS
PTF(K)=PTF(K)+PHI1(MN)*SN
AMX=AMX+MX(MN)*CS
AMTH=AMTH+MTH(MN)*CS
AMXTH=AMXTH+MXTH(MN)*SN
ANX=ANX+TX(MN)*CS
ANTH=ANTH+TH(MN)*CS
AQS=AQS+QS(MN)*CS
ANXTH=ANXTH+TXTH(MN)*SN
PF(K)=PF(K)+PHI1(MN)*SN
CCNTINUE
IF(K.EQ.1) WRITE(6,117)
117 IF(K.EQ.1.AND.IBCINL.LT.0).OR.(K.EQ.KMAX.AND.IBCFNL.LT.0))
1 GO TO 221
220 WRITE(6,118) K,ANX,ANTH,ANXTH,AMX,AMTH,AMXTH
221 GO TO 1354
1354 IF ($MODAL.OR.(ICHECK1.EQ.0)) GO TO 2
YNS(K)=ANX
YNTH(K)=ANTH
YQST(K)=ACS
YMS(K)=AMX
YMTH(K)=AMTH
YMSTH(K)=AMXTH
CCNTINUE
2 CCNTINUE
21 K=1,KMAX
FK=K-1
IFREQ=IFREQ
KTST=(K-1)/IFREQ
FKTST=FKTST
FKTST=FKTST/FIFREQ-FKTST
IF(K.EQ.1.OR.K.EQ.KMAX) GO TO 661
IF(FKTEST.NE.0) GO TO 658
IF(K.EQ.1) WRITE(6,217)
217 WRITE(6,218) K,X(1,K),X(2,K),X(3,K),X(4,K),PTF(K),PF(K)
IF ($MODAL.OR.(ICHECK2.EQ.0)) GO TO 658
YU(K)=X(1,K)
YV(K)=X(2,K)
YW(K)=X(3,K)
YPHS(K)=X(4,K)
YPHI(K)=PTF(K)
YPHI(K)=PF(K)
661

```



```

658 DO 659 I=1,4
659 X(I,K)=0.
660 CONTINUE
    IF ($PLOTS.AND..NOT.$MODAL.AND.((ICHCK1.GT.0).OR.(ICHCK2.GT.0)))
        1 CALL PLOT2(NTH)
    21 CONTINUE
    IF(IMODE.LE.0) RETURN
    DO 534 MN=1,MNMAX0
    IF(.NOT.$VIBES) WRITE(6,749) N(MN)
    DO 521 MM=1,MNMAX0
    I1=1+(MM-1)*KMAX2
    I2=I1+1
    U1(MM)=Z(1,I1)
    U2(MM)=Z(1,I2)
    V1(MM)=Z(2,I1)
    V2(MM)=Z(2,I2)
    W1(MM)=Z(3,I1)
    W2(MM)=Z(3,I2)
    521 CONTINUE
    DO 445 K=1, KMAX
    K1=K+1
    CALL BD3(K,BS,DB,DS,DD)
    IF(K.EQ.1.AND.IBCINL.LT.0) CALL POLE(K)
    IF(K.EQ.KMAX.AND.IBCFNL.LT.0) CALL POLE(K)
    TXZ=TX(MN)
    TTHZ=TTH(MN)
    TXTZ=TX(MN)
    AMXZ=MX(MN)
    AMTHZ=MTH(MN)
    AMXTZ=MXT(MN)
    CSZ=QS(MN)
    X(1,K)=PHIX(MN)
    X(2,K)=PHIT(MN)
    X(3,K)=PHI(MN)
    IF(K.EQ.1.AND.IBCINL.LT.0) GO TO 583
    IF(K.EQ.KMAX.AND.IBCFNL.LT.0) GO TO 583
    CALL PHIBET(K)
    DEX=DECMX(K)
    RRA=1./R(K)
    CX=CMXI(K)
    CT=QMT(K)
    GA=GAM(K)
    COXT=OX-CT
    GDOO=GA#DCXT
    CO2C=CO2#DS
    ENR=EN-RRR
    CALL TLOAD(K)

```

26850  
26860  
26870  
26880  
26890  
26900  
26910  
26920  
26930  
26940  
26950  
26960  
26970  
26980  
26990  
27000  
27010  
27020  
27030  
27040  
27050  
27060  
27070  
27080  
27090  
27100  
27110  
27120  
27130  
27140  
27150  
27160  
27170  
27180  
27190  
27200  
27210  
27220  
27230  
27240  
27250  
27260  
27270  
27280  
27290  
27300  
27310  
27320



```

TTS=TT(MN)*ALOAD
EX=(U3(MN)-U1(MN))*TCL I+OX*W2(MN)+ENL*OSE*(BX3(MN)+BE3(MN))
ET=ENR*V2(MN)+GA*U2(MN)+OT*W2(MN)+ENL*OSE*(BT3(MN)+BE3(MN))
EXT=.5*((V3(MN)-V1(MN))*TDL I-ENR*U2(MN)-GA*V2(MN)+ENL*SOE*BXT3(MN)
1)
KT=ENR*PHIT(MN)+GA*PHIX(MN)
KXT=.5*((ENR*(-PHIX(MN)-GA*W2(MN)+(W3(MN)-W1(MN))*TDL I)+GDO*V2(MN)
1) +OT*(V3(MN)-V1(MN))*TDL I-GA*PHIT(MN)-DCXT*PHI(MN))
TXZ=(BS*(EX+NU*ET)-TIS)*ABZ
TTHZ=(BS*(ET+NU*EX)-TIS)*ABZ
TXTZ=BS*DI*EXT*ABZ
MKI=KI+(MN-I)*KMAX2
AMXZ=Z(4,MKI)
AMTHZ=NU*AMXZ+DD2D*KT-DI*MT(MN)*ALOAD
AMXTZ=DS*DI*KXT
MKI1=MKI+1
MKK1=MKI-1
QSZ=SIGO*TKN*LAM2*(GA*AMXZ+(Z(4,MKK1)-Z(4,MKI))*TDL I+ENR*AMXTZ
1) -GA*AMTHZ)
AMXZ=AMXZ*ABZ3
AMTHZ=AMTHZ*ABZ3
AMXTZ=AMXTZ*ABZ3
X(1,K)=PHIX(MN)*ABZO
X(2,K)=PHIT(MN)*ABZC
X(3,K)=PHI(MN)*ABZO
CO 533 MF=1,NM*MAXO
U1(MM)=U2(MM)
U2(MM)=U3(MM)
V1(MM)=V2(MM)
V2(MM)=V3(MM)
W1(MM)=W2(MM)
W2(MM)=W3(MM)
FK=K-1
IFREQ=IFREQ
KTST=(K-1)/IFREQ
FKTST=KTST
FKTEST=FK/IFREQ-FKTST
IF(K.EQ.1.CR.K.EC.KMAX) GO TO 583
IF(FKTEST.NE.O.) GO TO 445
CONTINUE
IF(K.EQ.1) WRITE(6,117)
IF(K.EQ.1.AND.IBCINL.LT.O).OR.(K.EQ.KMAX.AND.IBCFNL.LT.O))
1 GO TO 4444
GO TO 4445
4444 WRITE(6,1181) K,TXZ,TTHZ,TXTZ,AMXZ,AMTHZ,AMXTZ
GO TO 4446
4445 WRITE(6,118) K,TXZ,TTHZ,TXTZ,QSZ,AMXZ,AMTHZ,AMXTZ
4446 IF (.NOT.$PLOTS.CR..NOT.$MODAL.OR.(ICCHK1.EQ.O)) GO TO 445
27330
27340
27350
27360
27370
27380
27390
27400
27410
27420
27430
27440
27450
27460
27470
27480
27490
27500
27510
27520
27530
27540
27550
27560
27570
27580
27590
27600
27610
27620
27630
27640
27650
27660
27670
27680
27690
27700
27710
27720
27730
27740
27750
27760
27770
27780
27790
27800

```



```

YNS(K)=TXZ
YNTH(K)=THZ
YNSTH(K)=TXTZ
YQS(K)=QSZ
YMS(K)=AMXZ
YMTH(K)=AMTHZ
YNSTH(K)=AMXTZ
445 CONTINUE
446 WRITE(6,217)
      DO 447 K=1,KMAX
      FK=K-1
      FIFREQ=IFREQ
      FKST=(K-1)/IFREQ
      FKSTEST=FK/FIFREQ-FKST
      IF(K.EQ.1.OR.K.EQ.KMAX) GO TO 593
      IF(FKSTEST.NE.0.) GO TO 447
      KZ=K+1+(MN-1)*KMAX2
      UP=Z(1,KZ)*ABZN
      VP=Z(2,KZ)*ABZN
      WP=Z(3,KZ)*ABZN
      WRITE(6,218) K,UP,VP,WP,X(1,K),X(2,K),X(3,K)
      IF(.NOT.$PLOTS.CR..NOT.$MODAL.OR.(ICHCK2.EQ.0)) GO TO 447
      YU(K)=UP
      YV(K)=VP
      YW(K)=WP
      YPHIS(K)=X(1,K)
      YPHIT(K)=X(2,K)
      YPHI(K)=X(3,K)
      447 CONTINUE
      IF ($PLOTS.AND.$MODAL.AND.((ICHCK1.GT.0).OR.(ICHCK2.GT.0)))
      1 CALL PLOT2(1)
534 CONTINUE
C *****
101 FORMAT('1.1',
101 THE LOAD STEP NUMBER IS 'I2',
101 THE SOLUTION CONVERGED IN 'I2',
217 ITERATIONS'////)
116 FORMAT('0.1',
116 THE SUMMED FORCES, MOMENTS, DISPLACEMENTS AN
116 ID ROTATIONS FOLLOW FOR THETA ='E15.6'////)
117 FORMAT('1.1',
117 STATION N THETA M THETA N STHETA M STHETA'//)
118 FORMAT('1X,I3.3X,7E16.4')
1181 FORMAT('1X,I3.3X,3E16.4')
1181 NOT COMPUTED 'I3E16.4'
1181 THE TIME IS 'F5
1181 FC MAT '1.1',
1181 THE TIME IN SECONDS
1181 1.2, CR 'E9.3',
1181 THE SOLUTION CONVERGED IN 'I2',
1181 ITE
217 FORMAT('1.1',
217 STATION PHI S
217 PHI THETA V PHI'//)
W

```









```

7 IF (IBSTIF.EQ.0) GO TO 8
  WRITE (6,1000)
  CALL PLOTT (XSTATN,YRSTIF,NGKMAX,0)
  WRITE (6,1006) GO TO 9
8 IF (IDSTIF.EQ.0) GO TO 9
  WRITE (6,1000)
  CALL PLOTT (XSTATN,YDSTIF,NGKMAX,0)
  WRITE (6,1007) GO TO 10
9 IF (IBBSTIF.EQ.0) GO TO 10
  WRITE (6,1000)
  CALL PLOTT (XSTATN,YBSTIF,NGKMAX,0)
  WRITE (6,1008) GO TO 11
10 IF (IDBSTIF.EQ.0) GO TO 11
  WRITE (6,1000)
  CALL PLOTT (XSTATN,YDBSTIF,NGKMAX,0)
  WRITE (6,1009) GO TO 11
11 IF (IPR.EQ.0) GO TO 11
  WRITE (6,1000)
  CALL PLOTT (XSTATN,YPR,NGKMAX,0)
  WRITE (6,1010) GO TO 12
12 IF (IPS.EQ.0) GO TO 12
  WRITE (6,1000)
  CALL PLOTT (XSTATN,YPS,NGKMAX,0)
  WRITE (6,1011) GO TO 13
13 IF (IPT.EQ.0) GO TO 13
  WRITE (6,1000)
  CALL PLOTT (XSTATN,YPT,NGKMAX,0)
  WRITE (6,1012) GO TO 14
14 IF (ITT.EQ.0) GO TO 14
  WRITE (6,1000)
  CALL PLOTT (XSTATN,YTT,NGKMAX,0)
  WRITE (6,1013) GO TO 15
15 IF (IMT.EQ.0) GO TO 15
  WRITE (6,1000)
  CALL PLOTT (XSTATN,YMT,NGKMAX,0)
  WRITE (6,1014) GO TO 16
16 IF (IDTT.EQ.0) GO TO 16
  WRITE (6,1000)
  CALL PLOTT (XSTATN,YDTT,NGKMAX,0)
  WRITE (6,1015) GO TO 17
17 IF (IDMT.EQ.0) GO TO 17
  WRITE (6,1000)
  CALL PLOTT (XSTATN,YDMT,NGKMAX,0)
  WRITE (6,1016)
  WRITE (6,1000)

```

```

17 RETURN
1000 FORMAT ('1')

```

\*\*\*\*\*

28750  
28760  
28770  
28780  
28790  
28800  
28810  
28820  
28830  
28840  
28850  
28860  
28870  
28880  
28890  
28900  
28910  
28920  
28930  
28940  
28950  
28960  
28970  
28980  
28990  
29000  
29010  
29020  
29030  
29040  
29050  
29060  
29070  
29080  
29090  
29100  
29110  
29120  
29130  
29140  
29150  
29160  
29170  
29180  
29190  
29200  
29210  
29220



```

1001 FORMAT ('0',T10,'RADIUS VS STATION')
1002 FORMAT ('0',T10,'GAMMA VS STATION')
1003 FORMAT ('0',T10,'OMEGA-THETA VS STATION')
1004 FORMAT ('0',T10,'DECEGAIN-SESS VS STATION')
1005 FORMAT ('0',T10,'3-STIFFNESS VS STATION')
1006 FORMAT ('0',T10,'DB-STIFFNESS VS STATION')
1007 FORMAT ('0',T10,'CORRIDOR-LOADING VS STATION')
1008 FORMAT ('0',T10,'INCIDENTAL-LOADING VS STATION')
1009 FORMAT ('0',T10,'CIRCUMFERENTIAL-LOADING VS STATION')
1010 FORMAT ('0',T10,'THERMAL-LOADING VS STATION')
1011 FORMAT ('0',T10,'THERMAL-LOADING VS STATION')
1012 FORMAT ('0',T10,'THERMAL-LOADING VS STATION')
1013 FORMAT ('0',T10,'THERMAL-LOADING VS STATION')
1014 FORMAT ('0',T10,'THERMAL-LOADING VS STATION')
1015 FORMAT ('0',T10,'THERMAL-LOADING VS STATION')
1016 FORMAT ('0',T10,'THERMAL-LOADING VS STATION')
C *****
END
29230
29240
29250
29260
29270
29280
29290
29300
29310
29320
29330
29340
29350
29360
29370
29380
29390
29400

```

```

C *****
SUBROUTINE PLOT2(NTH)
C *****
THIS SUBROUTINE CALLS PLOTTING ROUTINES FOR APPROPRIATE (USER
C *****
SPECIFIED) OUTPUT QUANTITIES
C *****
IMPLICIT LOGICAL*1 ($)
COMMON /IBL4/ KMAX,KL
COMMON /BL19/ TH(36)
COMMON /BLPLOT/
1 2 3
COMMON /BLPLT1/
1 2 3 4 5 6
COMMON /BLPLT1/
C *****
NGKMAX=-KMAX
IF ($MODAL) GO TO 121
IF (INS.EQ.0) GO TO 4
WRITE (6,1000)
IF (INS.GT.0) CALL PLOTIT (XSTATN,YNS,KMAX,0)
IF (INS.LT.0) CALL PLOTIT (XSTATN,YNS,NGKMAX,0)
WRITE (6,1001) TH(NTH)
4 IF (INTH.EQ.0) GO TC 5
C *****
29410
29420
29430
29440
29450
29460
29470
29480
29490
29500
29510
29520
29530
29540
29550
29560
29570
29580
29590
29600
29610
29620
29630
29640
29650
29660
29670
29680

```





```

WRITE (6,1000)
IF (INTH.GT.0) CALL PLOTIT (XSTATN,YNTH,KMAX,0)
IF (INTH.LT.0) CALL PLOTIT (XSTATN,YNTH,NGKMAX,0)
WRITE (6,1002) TH(NTH)
5 IF (INSTH.EQ.0) GO TO 6
WRITE (6,1000)
IF (INSTH.GT.0) CALL PLOTIT (XSTATN,YNSTH,KMAX,0)
IF (INSTH.LT.0) CALL PLOTIT (XSTATN,YNSTH,NGKMAX,0)
WRITE (6,1003) TH(NTH)
6 IF (IQS.EQ.0) GO TO 7
WRITE (6,1000)
IF (IQS.GT.0) CALL PLOTIT (XSTATN,YQS,KMAX,0)
IF (IQS.LT.0) CALL PLOTIT (XSTATN,YQS,NGKMAX,0)
WRITE (6,1004) TH(NTH)
7 IF (IMS.EQ.0) GO TO 8
WRITE (6,1000)
IF (IMS.GT.0) CALL PLOTIT (XSTATN,YMS,KMAX,0)
IF (IMS.LT.0) CALL PLOTIT (XSTATN,YMS,NGKMAX,0)
WRITE (6,1005) TH(NTH)
8 IF (IMTH.EQ.0) GO TO 9
WRITE (6,1000)
IF (IMTH.GT.0) CALL PLOTIT (XSTATN,YMTH,KMAX,0)
IF (IMTH.LT.0) CALL PLOTIT (XSTATN,YMTH,NGKMAX,0)
WRITE (6,1006) TH(NTH)
9 IF (IMSTH.EQ.0) GO TO 1211
WRITE (6,1000)
IF (IMSTH.GT.0) CALL PLOTIT (XSTATN,YMSTH,KMAX,0)
IF (IMSTH.LT.0) CALL PLOTIT (XSTATN,YMSTH,NGKMAX,0)
WRITE (6,1007) TH(NTH)
1211 IF (IU.EQ.0) GO TO 10
WRITE (6,1000)
IF (IU.GT.0) CALL PLOTIT (XSTATN,YU,KMAX,0)
IF (IU.LT.0) CALL PLOTIT (XSTATN,YU,NGKMAX,0)
WRITE (6,1010) TH(NTH)
10 IF (IV.EQ.0) GO TO 11
WRITE (6,1000)
IF (IV.GT.0) CALL PLOTIT (XSTATN,YV,KMAX,0)
IF (IV.LT.0) CALL PLOTIT (XSTATN,YV,NGKMAX,0)
WRITE (6,1009) TH(NTH)
11 IF (IW.EQ.0) GO TO 12
WRITE (6,1000)
IF (IW.GT.0) CALL PLOTIT (XSTATN,YW,KMAX,0)
IF (IW.LT.0) CALL PLOTIT (XSTATN,YW,NGKMAX,0)
WRITE (6,1008) TH(NTH)
12 IF (IPHS.EQ.0) GO TO 13
WRITE (6,1000)
IF (IPHS.GT.0) CALL PLOTIT (XSTATN,YPHIS,KMAX,0)
IF (IPHS.LT.0) CALL PLOTIT (XSTATN,YPHIS,NGKMAX,0)

```

29690  
29700  
29710  
29720  
29730  
29740  
29750  
29760  
29770  
29780  
29790  
29800  
29810  
29820  
29830  
29840  
29850  
29860  
29870  
29880  
29890  
29900  
29910  
29920  
29930  
29940  
29950  
29960  
29970  
29980  
29990  
30000  
30010  
30020  
30030  
30040  
30050  
30060  
30070  
30080  
30090  
30100  
30110  
30120  
30130  
30140  
30150  
30160





```

13 WRITE (6,1011) TH(NTH)
   IF (IPHIT.EQ.0) GO TO 14
   WRITE (6,1000)
   IF (IPHIT.GT.0) CALL PLOTIT (XSTATN,YPHIT,KMAX,0)
   IF (IPHIT.LT.0) CALL PLOTIT (XSTATN,YPHIT,NGKMAX,0)
   WRITE (6,1012) TH(NTH)
   GO TO 21
14 WRITE (6,1000)
   IF (IPHI.GT.0) CALL PLOTIT (XSTATN,YPHI,KMAX,0)
   IF (IPHI.LT.0) CALL PLOTIT (XSTATN,YPHI,NGKMAX,0)
   WRITE (6,1013) TH(NTH)
   RETURN
21 IF (INS.EQ.0) GO TO 15
121 WRITE (6,1000)
   IF (INS.GT.0) CALL PLOTIT (XSTATN,YNS,KMAX,0)
   IF (INS.LT.0) CALL PLOTIT (XSTATN,YNS,NGKMAX,0)
   WRITE (6,2001)
   GO TO 16
15 WRITE (6,1000)
   IF (INTH.EQ.0) CALL PLOTIT (XSTATN,YNTH,KMAX,0)
   IF (INTH.GT.0) CALL PLOTIT (XSTATN,YNTH,NGKMAX,0)
   IF (INTH.LT.0) CALL PLOTIT (XSTATN,YNTH,NGKMAX,0)
   WRITE (6,2002)
   GO TO 17
16 WRITE (6,1000)
   IF (INSTH.GT.0) CALL PLOTIT (XSTATN,YNSTH,KMAX,0)
   IF (INSTH.LT.0) CALL PLOTIT (XSTATN,YNSTH,NGKMAX,0)
   WRITE (6,2003)
   GO TO 18
17 WRITE (6,1000)
   IF (IQS.GT.0) CALL PLOTIT (XSTATN,YQS,KMAX,0)
   IF (IQS.LT.0) CALL PLOTIT (XSTATN,YQS,NGKMAX,0)
   WRITE (6,2004)
   GO TO 19
18 WRITE (6,1000)
   IF (IMS.GT.0) CALL PLOTIT (XSTATN,YMS,KMAX,0)
   IF (IMS.LT.0) CALL PLOTIT (XSTATN,YMS,NGKMAX,0)
   WRITE (6,2005)
   GO TO 22
19 WRITE (6,1000)
   IF (IMTH.GT.0) CALL PLOTIT (XSTATN,YMTH,KMAX,0)
   IF (IMTH.LT.0) CALL PLOTIT (XSTATN,YMTH,NGKMAX,0)
   WRITE (6,2006)
   GO TO 231
22 WRITE (6,1000)
   IF (IMSTH.GT.0) CALL PLOTIT (XSTATN,YMSTH,KMAX,0)
   IF (IMSTH.LT.0) CALL PLOTIT (XSTATN,YMSTH,NGKMAX,0)
   WRITE (6,2007)
   IF (IU.EQ.0) GO TO 23
231

```

30170  
30180  
30190  
30200  
30210  
30220  
30230  
30240  
30250  
30260  
30270  
30280  
30290  
30300  
30310  
30320  
30330  
30340  
30350  
30360  
30370  
30380  
30390  
30400  
30410  
30420  
30430  
30440  
30450  
30460  
30470  
30480  
30490  
30500  
30510  
30520  
30530  
30540  
30550  
30560  
30570  
30580  
30590  
30600  
30610  
30620  
30630  
30640



30650  
30660  
30670  
30680  
30690  
30700  
30710  
30720  
30730  
30740  
30750  
30760  
30770  
30780  
30790  
30800  
30810  
30820  
30830  
30840  
30850  
30860  
30870  
30880  
30890  
30900  
30910  
30920  
30930  
30940  
30950  
30960  
30970  
30980  
30990  
31000  
31010  
31020  
31030  
31040  
31050  
31060  
31070  
31080  
31090  
31100  
31110  
31120

```

WRITE (6,1000) CALL PLOTIT (XSTATN,YU,KMAX,0)
IF (IU.GT.0) CALL PLOTIT (XSTATN,YU,NGKMAX,0)
IF (IU.LT.0) CALL PLOTIT (XSTATN,YU,NGKMAX,0)
WRITE (6,2010) GO TO 24
23 WRITE (6,1000)
IF (IV.GT.0) CALL PLOTIT (XSTATN,YV,KMAX,0)
IF (IV.LT.0) CALL PLOTIT (XSTATN,YV,NGKMAX,0)
WRITE (6,2009) GO TO 25
24 WRITE (6,1000)
IF (IW.GT.0) CALL PLOTIT (XSTATN,YW,KMAX,0)
IF (IW.LT.0) CALL PLOTIT (XSTATN,YW,NGKMAX,0)
WRITE (6,2008) GO TO 26
25 IF (IPHS.EQ.0) GO TO 26
WRITE (6,1000) CALL PLOTIT (XSTATN,YPHIS,KMAX,0)
IF (IPHS.GT.0) CALL PLOTIT (XSTATN,YPHIS,NGKMAX,0)
IF (IPHS.LT.0) CALL PLOTIT (XSTATN,YPHIS,NGKMAX,0)
WRITE (6,2011) GO TO 27
26 IF (IPHT.EQ.0) GO TO 27
WRITE (6,1000) CALL PLOTIT (XSTATN,YPHIT,KMAX,0)
IF (IPHT.GT.0) CALL PLOTIT (XSTATN,YPHIT,NGKMAX,0)
IF (IPHT.LT.0) CALL PLOTIT (XSTATN,YPHIT,NGKMAX,0)
WRITE (6,2012) GO TO 28
27 IF (IPHI.EQ.0) GO TO 28
WRITE (6,1000) CALL PLOTIT (XSTATN,YPHI,KMAX,0)
IF (IPHI.GT.0) CALL PLOTIT (XSTATN,YPHI,NGKMAX,0)
IF (IPHI.LT.0) CALL PLOTIT (XSTATN,YPHI,NGKMAX,0)
WRITE (6,2013)
28 RETURN

```

```

C*****
1000 FORMAT ('1',T10,'S'-MEMBRANE FORCE VS STATN, MERIDIAN AT THETA =
1001 1,F10.5,RADIANS')
1002 1,F10.5,RADIANS')
1003 1,F10.5,RADIANS')
1004 1,F10.5,RADIANS')
1005 1,F10.5,RADIANS')
1006 1,F10.5,RADIANS')
1007 1,F10.5,RADIANS')
1008 1,F10.5,RADIANS')

```









```

XMAX=X(I)
6 IF(X(I).GT.XMIN) GO TO 7
XMIN=X(I)
7 IF(Y(I).LT.YMAX) GO TO 8
YMAX=Y(I)
8 IF(Y(I).GT.YMIN) GC TO 1
YMIN=Y(I)
1 CCNTINUE
C *****
C IF NOT AUTOSCALE GO TO CALL DRAWIT
C *****
C IF(N.GT.0) GO TO 5
C *****
C COMPUTE X-SCALE & NEW XMAX AND XMIN
C *****
C CALL SCALIT(XMAX,XMIN,4)
C *****
C COMPUTE Y-SCALE & NEW YMAX AND YMIN
C *****
C CALL SCALIT(YMAX,YMIN,6)
C *****
C PLOT CURVE
C *****
5 CALL DRAWIT(X,Y,KN,RANGE,S,1,MODCUR)
IF(MODCUR.EQ.1.CR.MODCUR.EQ.2) RETURN
C *****
C PRINT SCALES WHEN LAST CURVE PLOTTED
C *****
XS=(XMAX-XMIN)/80.
YS=(YMAX-YMIN)/60.
WRITE(6,100) XS,YS
100 FORMAT(15X,'X-SCALE: ',E10.3,'Y-SCALE: ',E10.3,' UNITS',//
1 RETURN
C *****
END
C *****

SUBROUTINE SCALIT(XMAX,XMIN,IDIV)
C *****
C ROUNO MAXIMUM TO NEXT HIGHEST 2 SIG FIGS
C *****
DIV=IDIV
XMAX=AMAX1(0.,XMAX)
CALL ROUND(XMAX,IMX,FMX)
IMX=IMX-1
FMX=FMX*10.
3 XMX=FMX*10.

```

```

31590
31600
31610
31620
31630
31640
31650
31660
31670
31680
31690
31700
31710
31720
31730
31740
31750
31760
31770
31780
31790
31800
31810
31820
31830
31840
31850
31860
31870
31880
31890
31900
31910
31920
31930
31940

31950
31960
31970
31980
31990
32000
32010
32020
32030
32040

```









32530  
32540  
32550  
32560  
32570  
32580  
32590  
32600  
32610  
32620  
32630  
32640  
32650  
32660  
32670  
32680  
32690  
32700  
32710  
32720  
32730

```

10 IF (DIV.GT.4) GO TO 15
C IF X SCALE BETWEEN 8. AND 10. ROUNDED TO 10.
C
FFX=ABS(FACTX/10.)
IF (FFX.GT.8. AND FFX.LT.10.) FFX=10.
IF (FACTX.LT.0.) FFX=-10.
FACTX=FFX
15 XSC=FACTX/10.
C COMPUTE NEW MAX & MIN FROM ROUNDED SCALE
C
IF (XM.NE.0.) GO TO 4
IF (XM.NE.0.) XMIN=-XSC
IF (XM.NE.0.) XMAX=XSC
RETURN
4 XMAX=XSC+XMN
XMIN=XMN
RETURN
END

```

32740  
32750  
32760  
32770  
32780  
32790  
32800  
32810  
32820  
32830  
32840  
32850  
32860  
32870  
32880  
32890  
32900  
32910  
32920  
32930  
32940  
32950  
32960  
32970  
32980

```

SUBROUTINE ROUNDA(ANUM,IS,FACT)
C EXPRESS ANUM IN SCIENTIFIC NOTATION WHEN
C ANUM=FACT*10. IS BETWEEN 1. AND 9.9
C
IF (ANUM.EQ.0.) GO TO 15
BNUM=ANUM
IF (BNUM.LT.0.) BNUM=-BNUM
IS=ALOG(BNUM)*.43429448
FACT=BNUM/10.
C FIND POWER OF 10
IDC=-3
R2=0
DO 10 I=1,5
IDC=IDC+1
R1=R2
R2=10.* (IDC+1)
IF (FACT.GE.R1 AND FACT.LT.R2) GO TO 8
10 CONTINUE
8 FACT=FACT*10.* (-IDC)
IS=IS+IDC
C ROUND MANTISSA TO 2 SIG FIGS

```



```

C ***
C *** IFAC=FACT*10.+-.05
C *** FACT=IFAC
C *** FACT=FACT/10.
C *** IF(FACT.LT.10.) GO TO 20
C *** SET TO 1 IF LESS THAN 10.
C *** FACT=1.
C *** IS=IS+1
C *** IF INPUT NEGATIVE, SET PANTISSA NEGATIVE
C *** 20 IF(ANUM.LT.0.) FACT=-FACT
C *** RETURN
C *** SET TO 0. IF 0.
C *** 15 FACT=0.
C *** IS=0
C *** RETURN
C *** END

C ***
C *** SUBROUTINE DRAWIT(X,Y,ICATA,RANGES,KKZ,MODCUR)
C *** DIMENSION GRID(61,81),XSCALE(5),YSCALE(7)
C *** DIMENSION X(1),Y(1),RANGES(4)
C *** INTEGER*2 GRID,BLANK,DOT,XCHAR(4)/1H+,1H-,1H.,1HX/
C *** DATA DOT,BLANK/Z4B4J,Z4J4J/
C *** KCATA=NCATA*KKZ
C *** IF(MODCUR.GT.1) GO TO 444
C *** GRID IS THE MATRIX USED TO PLOT THE POINTS
C *** IERR=J
C *** XMAX = RANGES(1)
C *** XMIN = RANGES(2)
C *** YMAX = RANGES(3)
C *** YMIN = RANGES(4)
C *** CHECKING X AND Y POINTS AND PLOTTING THOSE CUT OF RANGE
C *** AT THE MARGIN
C *** DO 30 I=1,KCATA*KKZ
C *** IF(X(I).GT.XMAX.OR.X(I).LT.XMIN.OR.Y(I).GT.YMAX.OR.Y(I).LT.YMIN)
32990
33000
33010
33020
33030
33040
33050
33060
33070
33080
33090
33100
33110
33120
33130
33140
33150
33160
33170
33180
33190
33200
33210

33220
33225
33230
33240
33250
33260
33265
33270
33280
33290
33300
33310
33320
33330
33340
33350
33360
33370
33380
33390
33400
33410
33420

```





3430  
3440  
3450  
3460  
3470  
3480  
3490  
3500  
3510  
3520  
3530  
3540  
3550  
3560  
3570  
3580  
3590  
3600  
3610  
3620  
3630  
3640  
3650  
3660  
3670  
3680  
3690  
3700  
3710  
3720  
3730  
3740  
3750  
3760  
3770  
3780  
3790  
3800  
3810  
3820  
3830  
3840  
3850  
3860  
3870  
3880  
3890  
3900

```

1  IERR=IERR+1
  IF(X(I).LE.XMAX) GO TO 205
  X(I)=XMAX
  GO TO 210
205 IF(X(I).GE.XMIN) GO TO 210
  X(I)=XMIN
210 IF(Y(I).LE.YMAX) GO TO 215
  Y(I)=YMAX
  GO TO 30
215 IF(Y(I).GE.YMIN) GO TO 30
  Y(I)=YMIN
30 CONTINUE
C** PLOTTING X AND Y AXIS, IF NECESSARY
C**
C** JERR=0
  X RANGE=XMAX-XMIN
  Y RANGE=YMAX-YMIN
  IF (Y RANGE.NE.0.) GO TO 298
  IF(YMIN.EQ.0.) GO TO 889
  YMIN=J
  Y RANGE=YMAX
  GO TO 299
298 IF (X RANGE.NE.0.) GC TO 299
  IF(XMIN.EQ.0.) GO TO 887
  XMIN=0
  X RANGE=XMAX
  BLANKING CUT MATRIX-(GRID)
C**
C** 299 DO 300 I=1,61
  DO 301 JJ=1,81
    GRID(I,JJ)=BLANK
  CONTINUE
300 IF(XMAX* XMIN.GE.0.) GO TO 222
  IYAXIS=80./(-XMIN)/X RANGE+1.5
  DO 40 I=1,61
    GRID(I,IYAXIS)=DCT
  40 GRID(I,IYAXIS)=DCT
  222 IF(YMAX* YMIN.GE.0.) GO TO 333
  IXAXIS=60./YMAX/Y RANGE+1.5
  DO 60 I=1,81
    GRID(IXAXIS,I)=DCT
  60 GRID(IXAXIS,I)=DCT
C**
C** COMPUTE PROPER SCALE NUMBERS
C**
333 XINCR=X RANGE/4.
  YINCR=Y RANGE/6.
  XSCALE(1)=XMAX

```









```

18  FORMAT(3X,1PE10.3,2X,1H+,1X,81A1,1X,1H+,2X,E10.3)
34390
GC TO 405
34400
404  WRITE(6,118) YSCALE(11),(GRID(1K,IX),IX=1,81),YSCALE(11)
34410
118  FORMAT(2X,F11.2,'+',81A1,'+',F11.2)
34420
405  II=II+1
34430
GO TO 101
34440
92  WRITE(6,19) (GRID(1K,IX),IX=1,81)
34450
19  FORMAT(15X,'+',81A1,'+')
34460
101  CONTINUE
34470
22  IF(AXR.LT.1.E+8.AND.AXR.GE..95) GO TO 402
34480
WRITE(6,22) XSCALE
34490
FORMAT(15X,'+',81'*****'),'*****'/12X,1PE10.3,4(10X,E10.3),//)
34500
GO TO 403
34510
402  WRITE(6,217) XSCALE
34520
217  FORMAT(15X,'+',81'*****'),'*****'/ 8X,F11.2,4( 9X,F11.2),//)
34530
403  IF(IERR.GT.0) WRITE(6,20) IERR
34540
20  FORMAT(10X,'NUMBER OF POINTS OUT OF RANGE =',I4)
34550
1000 RETURN
34560
889 WRITE(6,888)
34570
888  FORMAT(' ALL Y VALUES=0. CANNOT SETUP PLOT GRID. CHECK MAX & MIN Y
34580
1 WHEN MODCUR=0 CR 1.')
34590
JERR=10
34600
RETURN
34610
887 WRITE(6,886)
34620
886  FORMAT(' ALL X VALUES=0. CANNOT SETUP PLCT GRID. CHECK MAX & MIN X
34630
1 WHEN MODCUR=0 OR 1.')
34640
JERR=10
34650
RETURN
34660
885 WRITE(6,884)
34670
884  FORMAT(' GRID NOT SETUP WHEN MODCUR LAST 0 OR 1. NO PLOT UNTIL GRI
34680
10 PROPERLY SETUP.')
34690
RETURN
34700
END
34710

```



```

SUBROUTINE RANGER (*)
C *****
C SUBROUTINE RANGER IS THE CONTROLLING SUBROUTINE FOR THE FREQUENCY
C RANGE SEARCH OPERATION. THE TOP AND BOTTOM OF THE RANGE IS CON-
C VERTED FROM CPS TO RADIANS/SEC AND A CHECK PERFORMED ON THE
C LIMITS. THE LOWEST (SUB. BTM) AND HIGHEST (SUB. TOPPER)
C FREQUENCIES ARE DETERMINED PRIOR TO THE RANGE BEING PARTITIONED.
C THE ALGORITHM IS BIAISED TOWARD THE LOWER FREQUENCIES. A FORM OF
C RATE-AITKIN EXTREME ADJUST IS USED TO INCREASE THE CONVERGENCE
C PARTICULAR CASE.
C *****
C IMPLICIT LOGICAL I(*)
C REAL*8 ZERO, PTFIVE, CNEDO, SMALL, VREAL, VIMAG, EPSVIB, EPSVEC, EPSAIT,
9 SCALEW, SCALEX, SCALE3, VR, VI, WW, XX, YY, XAITKN, VALNEW, VALOLD, PCNT,
8 TWOPI, WV, XV, YV
C COMMON /BL24/ DG(4,4), RANGE(100,4), TOP, BOTTOM, DELSHF, TOLSHF, RZ(12)
C COMMON /BL100/ TEE0, $DYNMC
C COMMON /BL104/ WV(4,203), XV(4,203), YV(4,203), FILL(28), SHFTPT(3,100)
C COMMON /BLK2/ IVIBS, IVB
C COMMON /BLK3/ TWOPI, IFEVIN, IENDO, SMALL, VREAL, VIMAG, AREAL
C COMMON /BLK4/ ZERO, PTFIVE, ONEC
C COMMON /BLK5/ $FLAG, $TAG, $DOVEC
C COMMON /BLK6/ NJ, NK, I4, IRGOLD, NUMBER, ITWO
C COMMON /BLK7/ SCALEX, SCALEW, SCALE3
C COMMON /BLK8/ EPSVIB, EPSVEC, EPSAIT
C COMMON /BLK9/ ITR1, ITR2
C COMMON /BLK10/ ASHIFT, ASHFUM, ATSHF, PI1, TOLUP, TOLDWN, TOLCLS
C COMMON /BLK13/ DIF1, DIF2, DIF3, DIF4, TOLLOW, TOLHI, TOLRNG
C XAITKN(WW,XX,YY) = WV - (((WW-XX)*2)/((WW+YY)-(2.0*XX)))
C *****
C BOTTOM = -((SHFTPT(1,IVB) * SNGL(TWOPI) * TEE0)**2)
C TOP = -((SHFTPT(2,IVB) * SNGL(TWOPI) * TEE0)**2)
C CALL VECOUT(1, &1)
C CALL VECOUT(1, &1)
1 IF(BOTTOM.LE.-TOP) CALL VECOUT(4, &9)
C CALL BOTTOM(&9)
C CALL TOPPER(&9)
C IRGOLD = 1
2 IF(IRGOLD.LE.0) CALL VECOUT(19, &9)
I4 = 1
VALNEW = ZERO
$TAG = .FALSE.
IF(RANGE(I4,2).LE.(TOLRNG*RANGE(I4,3))) CALL ADJUST(1,&2,&2)
$TAG = .TRUE.
DEL R = (RANGE(I4,3) - RANGE(I4,2)) * TOLCLS
ASHIFT = RANGE(I4,2) +

```





```

DELSHF = RANGE(I4,3) - DELR
WRITE(6,333) I4,(RANGE(I4,J),J=1,4),(RANGE(I4+1,K),K=1,4)
FORMAT(/,/35X,'RANGE',I2,'=',4E14.6//35X,'RANGE 2=',4E14.6)
CALL STARTX(2)
CALL PMATRIX
DO 4 I5 = 1,ITR1
$FLAG = .TRUE.
CALL ITRATE
VALCLO = VALNEW
VALNEW = XAITKN(SCALEW,SCALEX,SCALE3)
PCNT=DABS((DABS(VALNEW)-DABS(VALCLO))/VALNEW)
IF(NJ.GT.I.TR2) CALL VECOUT(7,&4)
IF(PCNT.LT.EPSAIT.AND.NJ.GT.3) GO TO 8
CONTINUE
CALL VECOUT(8,&5)
CALL ITCPLX(&6)
IF($FLAG) CALL ADJUST(5,&7,&2)
CALL ADJUST(1,&7,&2)
$TAG = .FALSE.
ASHIFT = DELSHF
GO TO 3
DIF1 = TOLLOW * RANGE(I4,1)
DIF2 = TOLHI * RANGE(I4,1)
VREAL = CHEDD / VALNEW
VREAL = DBLE(ASHFAL)
AREAL = SNGL(VREAL)
IF(AREAL.LE.DIF1.AND.AREAL.GE.DIF2) CALL ADJUST(2,&7,&2)
DIF3 = TOLLOW * RANGE(I4,4)
DIF4 = TOLHI * RANGE(I4,4)
IF(AREAL.LE.DIF3.AND.AREAL.GE.DIF4) CALL ADJUST(3,&7,&2)
IF(AREAL.LE.DIF2.AND.AREAL.GE.DIF3) CALL ADJUST(4,&7,&2)
IF(AREAL.GT.DIF1.OR.AREAL.LT.DIF4) CALL ADJUST(1,&7,&2)
RETURN
END

```

35180  
35190  
35200  
35210  
35220  
35230  
35240  
35250  
35260  
35270  
35280  
35290  
35300  
35310  
35320  
35330  
35340  
35350  
35360  
35370  
35380  
35390  
35400  
35410  
35420  
35430  
35440  
35450  
35460  
35470  
35480  
35490  
35500  
35510  
35520  
35530

```

SUBROUTINE ADJUST(IADJ,*)
THIS SUBROUTINE, CALLED BY SUBROUTINE RANGER, PERFORMS THE FINAL
DISPOSITION OF EACH SEARCH VECTOR WHEN A NEW RECKED EIGENVALUE
IS DETERMINED (IADJ=7). IT IS REFINED, RECKED TO ENSURE
THAT IT IS IN THE RANGE OF THE EIGENVALUE FALLS
OUT OF THE RANGE, OR A SUITABLE POSITION THEREOF, IS ELIMINATED.
THE TOTAL RANGE, OR A SUITABLE POSITION THEREOF, IS ELIMINATED.

```

5540  
5550  
5560  
5570  
5580  
5590  
5600  
5610  
5620  
5630





```

IMPLICIT LOGICAL*1 ($)
REAL*8 ZERO,CO,P,FIVE,ONE,FO,SMALL,VREAL,VIMAG
COMMON/BLK24/DG(4,4,4),RANGE(IJJ,4),TOP,BOTTOM,DELSHF,TOLSHF,RZ(12)
COMMON/BLK4/ZERO,CO,P,FIVE,ONE,FO,SMALL,VREAL,VIMAG,AREAL
COMMON/BLK5/$FLAG,$TAG,$DOVEC
COMMON/BLK6/NJ,NK,I4,IRGOLD,NUMBER,ITWO
COMMON/BLK10/ASHIFT,ASHFUM,ATSHF,BIT,TOLUP,TOLDWN,TOLCLS
COMMON/BLK13/DIF1,DIF2,DIF3,DIF4,TOLLOX,TOLHI,TOLRNG
C*****
GO TO (1,3,4,7,12),IADJ
1 IF($TAG) RETURN 1
5 DO 2 J = 1,4
DO 2 I = 1,4,IRGOLD
IPI = I + 1
2 RANGE(I,J) = RANGE(IPI,J)
IRGOLD = IRGOLD - 1
RETURN 2
3 RANGE(I4,2) = ASHFUM + (ASHFUM - RANGE(I4,1))
IF($TAG) RETURN 1
GO TO 6
4 RANGE(I4,3) = ASHFUM - (RANGE(I4,4) - ASHFUM)
6 IF(RANGE(I4,2).LE.(TOLRNG*RANGE(I4,3))) GO TO 5
IF($TAG) RETURN 1
RETURN 2
7 ASHFUM = ASHFUM + TOLSHF * (AREAL - ASHFUM)
ARNG1 = ASHFUM
CALL PMATRX
CALL ITREAL(E8)
CALL ITCPLX(E1)
GO TO 1
8 ASHFUM = ARNG1
DIF1 = TOLLOX * RANGE(I4,1)
DIF2 = TOLHI * RANGE(I4,1)
DIF3 = TOLLOX * RANGE(I4,4)
DIF4 = TOLHI * RANGE(I4,4)
IF(AREAL.LE.DIF1.AND.AREAL.GE.DIF2) GO TO 3
IF(AREAL.LE.DIF3.AND.AREAL.GE.DIF4) GO TO 4
IF(AREAL.GT.DIF1.OR.AREAL.LT.DIF4) GO TO 1
CALL VECOUT(9,E9)
GO TO 1
9 IM1 = IRGOLD + 2
DO 10 J = I4,IRGOLD
DO 10 I = IM1 - 1
IM2 = IM1 - 1
10 RANGE(IM1,I) = RANGE(IM2,I)
IRGOLD = IRGOLD + 1
NUMBER = NUMBER + 1
RANGE(I4,3) = AREAL

```



```

RANGE(I4,4) = AREAL
RANGE(I4+1,1) = AREAL
RANGE(I4+1,2) = AREAL
IF(RANGE(I4,2).LE.(TOLRNG*RANGE(I4,3))) GO TO 5
GO TO (11,14,16),ITWO
11 RETURN 2
12 DIF1 = TOLLOW * RANGE(I4,1)
DIF2 = TOLLOW * RANGE(I4,1)
DIF3 = TOLLOW * RANGE(I4,4)
DIF4 = TOLLOW * RANGE(I4,4)
AREAL = SNGL(VREAL)
AREAL = SNGL(VIMAG)
IF((BREAL.GE.DIF4.AND.BREAL.LE.DIF3).AND.(AREAL.GE.DIF2.AND.AREAL.
1 LE.DIF1)) GO TO 5
IF(BREAL.GE.DIF4.AND.BREAL.LE.DIF3) RANGE(I4,3) = ASHFUM
IF(AREAL.GE.DIF2.AND.AREAL.LE.DIF1) RANGE(I4,2) = ASHFUM
IF((AREAL.GT.DIF1.CR.AREAL.LT.DIF4).AND.(BREAL.LT.DIF4.OR.BREAL.GT
1 DIF1)) GO TO 1
IF((AREAL.LT.DIF2.AND.AREAL.GT.DIF3).AND.(BREAL.LT.DIF2.AND.BREAL
1 GT.DIF3)) GO TO 13
IF((AREAL.LT.DIF4.OR.AREAL.GT.DIF1).AND.(BREAL.GT.DIF3.AND.BREAL.L
1 T.DIF2)) GO TO 18
IF((BREAL.LT.DIF4.OR.BREAL.GT.DIF1).AND.(AREAL.GT.DIF3.AND.AREAL.L
1 T.DIF2)) GO TO 17
13 RETURN 2
DIST1 = ABS(ASHFUM - BREAL)
DIST2 = ABS(AREAL - ASHFUM)
ATEMP = AREAL
IF(DIST2.GT.DIST1) GO TO 15
ITWO = 2
GO TO 7
14 AREAL = BREAL
I4 = I4 + 1
CALL START(2)
GO TO 7
15 AREAL = BREAL
ITWO = 3
GO TO 7
16 AREAL = ATEMP
CALL START(2)
GO TO 7
17 ITWO = 1
GO TO 7
18 AREAL = BREAL
ITWO = 1
GO TO 7
END

```



```

C** SUBROUTINE BTIM(*)
C**
C** SUBROUTINE BTIM DETERMINES THE LOWEST FREQUENCY IN THE OVERALL
C** RANGE OF THE CONVERGENCE TEST ON THE VECTOR IS ALWAYS PERFORMED.
C** A FORM OF THE AKIN EXTRAPOLATION IS USED TO INCREASE THE CON-
C** VERGENCE RATE. THE RANGE WILL BE SEARCHED BOTTOM TO TOP IN PRE-
C** DETERMINED SEGMENTS UNTIL A REAL EIGENVALUE IS DETERMINED AS
C** THE BOTTOM (TOLUP AND NTRY IN SUBROUTINE INITAA).
C**
C** IMPLICIT LOGICAL*1 ($)
C** REAL*8 ZERO, PTFIVE, ONEDO, SMALL, VREAL, VIMAG, EPSVIB, EPSAIT,
C** 9 SCALEW, SCALEX, SCALES, VR, VI, WW, XX, YY, XAI, TKN, TEMPI, TEMP2,
C** 8 TWOPI, CLDVR
C** COMMON/BLK24/DG(4,4,4), RANGE(100,4), TOP, BOTTOM, DELSHF, TOLSHF, RZ(12)
C** COMMON/BLK4/ZEROCO, PTFIVE, ONEDO, SMALL, VREAL, VIMAG, AREAL
C** COMMON/BLK5/$FLAG, $TAG, $DOVEC
C** COMMON/BLK6/NJ, NK, I4, IRGOLD, NUMBER, ITWO
C** COMMON/BLK7/SCALEX, SCALEW, SCALE3
C** COMMON/BLK8/EPSVIB, EPSVEC, EPSAIT
C** COMMON/BLK9/ITR1, ITR2
C** COMMON/BLK10/ASHIFT, ASHFUM, ATSHF, BII, TOLUP, TOLDWN, TOLCLS
C** COMMON/BLK12/NTRY
C** SIZE(VR, VI) = - SNGL(DSORT(VR*VR + VI*VI))
C** XAI, TKN(WW, XX, YY) = EW - (((WW-XX)/((WW + YY)-(2.0 * XX)))
C**
C** $TAG = $DOVEC
C** $DOVEC = .TRUE.
C**
C** NI = 1
C** DELSHF = TOLUP * (TOP - BOTTOM)
C** DO 8 I = 1, 4
C** 1 RANGE(2, I) = 0.0
C** IF (BOTTOM.LE.TOP) CALL VECOUT(11, &5)
C** IF (NTRY) CALL VECOUT(12, &5)
C** IF (NI.GT.1)
C** CALL START(1)
C** ASHIFT = BOTTOM
C** CALL PMATX, ITR1
C** DO 9 I = 1, ITR1
C** $FLAG ITRATE
C** CALL ITRATE
C** TKN(SCALEW, SCALEX, SCALE3)
C** TEMP1 = XAIS(TEMP1)
C** TEMP2 = CABS(TEMP1)
C** TEMP3 = CABS((TEMP2 - CLDVR)/TEMP2)
C** CLDVR = TEMP2
C** IF (NJ.GT.1) CALL VECOUT(7, &9)
C** IF (TEMP3.GE.EPSAIT)
C** IF (NJ.LT.3) GO TO 9
C** VREAL = (ONEDO/TEMP1) + DBLE(ASHFUM)

```





```

11 11 ASHIFT = ASHFUM + TOLSHF * (SNGL(VREAL) - ASHFUM)
    IF (ABS(ASHIFT).LE.1.0) GO TO 11
    CALL PMATRIX
    CALL ITREAL(86)
    GO TO 11
19 19 CONTINUE
    CALL VECOUT(8,&10)
10 10 ARTSIZ = 0.0
    CALL ITCLPX(82)
    IF ($FLAG) GO TO 6
    ARTSIZ = SIZE(VREAL,VIMAG)
    BOTTOM = BOTTOM + DELSHF
2 2 IF (ARTSIZ.LT.BOTTCM) BOTTOM = BOTTOM - DELSHF + 1.1 * ARTSIZ
    CALL VECOUT(10,&1)
6 6 AREAL = SNGL(VREAL)
    AIMAG = SNGL(VIMAG)
    DIST1 = ABS(AIMAG - BOTTOM)
    DIST2 = ABS(AREAL - BOTTOM)
    IF (DIST1.LT.DIST2.AND.VIMAG.LT.ZERODO) AREAL = AIMAG
    NUMBER(1,1) = AREAL
    $DOVEC = 1
    IF (AREAL.LE.BOTTCM) RANGE(1,2) = AREAL
    IF (AREAL.GT.BOTTCM) RANGE(1,2) = BOTTOM
    CALL VECOUT(20,&3)
3 3 IF (VIMAG.EQ.ZERODO) CALL VECOUT(9,&7)
7 7 IF (RANGE(1,2).LE.TOP) CALL VECOUT(11,&5)
5 5 RETURN
    END

```

```

37070
37080
37090
37100
37110
37120
37130
37140
37150
37160
37170
37180
37190
37200
37210
37220
37230
37240
37250
37260
37270
37280
37290
37300
37310
37320
37330
37340
37350
37360

```

```

C ***** SUBROUTINE TOPPER(*) *****
C ***** SUBROUTINE TOPPER DETERMINES THE HIGHEST FREQUENCY IN THE OVERALL *****
C ***** RANGE. IT OPERATES IN A MANNER SIMILAR TO SUBROUTINE BTM. *****
C ***** IMPLICIT LOGICAL*11 ($) *****
    REAL*8 ZERODO,PTFIVE,QNVED3,SMALL,VREAL,VIMAG,EPVIB,EPVSEC,EPVSAIT,
9 SCALEX,SCALE3,VX,VI,VW,XX,YY,XAITKN,TEMP1,TEMP2,TEMP3,
8 TWOPI,OLDVR
    COMMON/BLK24/DG(4,4,4),AGE(100,4),TOP,BOTTOM,DELSHF,TOLSHF,RZ(12)
    COMMON/BLK4/ZERODO,BTTCM,CNVED3,PTTCM,DOVEC
    COMMON/BLK5/$FLAG,$TAG,$DOVEC
    COMMON/BLK6/NJ,NK,I4,NGOLD,NUMBER,ITWO
    COMMON/BLK7/SCALEX,SCALE3
    COMMON/BLK8/EPVIB,EPVSEC,EPVSAIT
    COMMON/BLK9/ITR1,ITR2

```

```

37370
37380
37390
37400
37410
37420
37430
37440
37450
37460
37470
37480
37490
37500
37510
37520

```





```

COMMON/BLK13/ASHIFT,ASHFUM,ATSF,F,BII,TOLUP,TOLDWN,TOLCLS
COMMON/BLK12/NTRY
SIZE(VR,VI) = - SNGL(DSORT(VR*VR + VI*VI))
XAITKN(WW,XX,YY) = WW - ((WW-XX)**2)/((WW + YY)-(2.0 * XX)))
C** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** **
IF(TOP .EQ. 0.0) TOP = -1000.0
NI = 1
1 IF(RANGE(1,2),LE.TOP) CALL VECOUT(13,&5)
IF(NI.GT.NTRY) CALL VECOUT(14,&5)
NI = NI + 1
CALL START(2)
ASHIFT = TOP
CALL PMATRIX
DO 9 I = 1,ITR1
$FLAG = .TRUE.
CALL ITRATE
TEMP1 = XAITKN(SCALEW,SCALEX,SCALE3)
TEMP2 = DABS(TEMP1)
TEMP3 = DABS(TEMP2 - OLDVR)/TEMP2
OLDVR = TEMP2
IF(NJ.GT.ITR2) CALL VECOUT(7,&9)
IF(TEMP3.GE.EP.SAIT) GO TO 9
IF(NJ.LT.3) GO TO 9
VREAL = (ONEDO/TEMP1) + DBLE(ASHFUM)
ASHIFT = ASHFUM + TOLSHF * (SNGL(VREAL) - ASHFUM)
CALL PMATRIX
CALL ITRREAL(&6)
GO TO 10
CONTINUE
9 CALL VECOUT(8,&10)
10 ARTSIZ = 0.0
CALL ITCPLX(&2)
IF($FLAG) GO TO 6
ARTSIZ = SIZE(VREAL,VIMAG)
2 TOP = TOP - DELSHF
TOP = TOP + DELSHF - 1.1 * ARTSIZ
6 CALL VECOUT(15,&1)
AREAL = SNGL(VREAL)
AIMAG = SNGL(VIMAG)
DIST1 = ABS(AIMAG - TOP)
DIST2 = ABS(AREAL - TOP)
IF(DIST1.LT.DIST2.AND.VIMAG.LT.ZERODO) AREAL = AIMAG
NUMBER = 2
IF(AREAL.GT.TOP) RANGE(1,3) = AREAL
IF(AREAL.LE.TOP) RANGE(1,3) = TOP
CALL VECOUT(21,&3)
3 IF(VIMAG.EQ.ZEROCC) CALL VECOUT(9,&7)
37530
37540
37550
37560
37570
37580
37590
37600
37610
37620
37630
37640
37650
37660
37670
37680
37690
37700
37710
37720
37730
37740
37750
37760
37770
37780
37790
37800
37810
37820
37830
37840
37850
37860
37870
37880
37890
37900
37910
37920
37930
37940
37950
37960
37970
37980
37990
38000

```



```

7 IF(RANGE(1,2).LE.RANGE(1,3)) CALL VECOUT(11,55)
4 RETURN
5
END

```

```

38010
38020
38030
38040

```

```

C** SUBROUTINE ITRREAL(*)
C** THIS SUBROUTINE DIRECTS THE ITERATION PROCESS THAT DETERMINES
C** REAL EIGENVALUES AND VECTORS. THE CONVERGENCE TEST ON THE EIGEN-
C** VALUES IS PERFORMED IN THIS SUBROUTINE.
C** IMPLICIT LOGICAL I(*)
C** REAL*8 ZERO,PTFIVE,CNEDO,SMALL,VREAL,VIMAG,EPSPVIB,EPSPVIB,EPSPVIB,EPSPVIB,
1 SCALEW,SCALEX,SCALE3,CNEDVR,OLDVR,TEMP,WV,XV,YV
COMMON /I8L12/ KMAX1,KMAX2,NCCNV
COMMON /BLK1/ VMAS,NM2,$VIBES
COMMON /BLK4/ ZEROC,PTFIVE,CNEDO,SMALL,VREAL,VIMAG,AREAL
COMMON /BLK5/ $FLAG,$TAG,$DOVEC
COMMON /BLK6/ NJ,NK,I4,IRGCLOD,NUMBER,ITWO
COMMON /BLK7/ SCALEX,SCALEW,SCALE3
COMMON /BLK8/ EPSPVIB,EPSPVIB,EPSPVIB,EPSPVIB,EPSPVIB,EPSPVIB
COMMON /BLK9/ ITR1,ITR2
COMMON /BLK10/ ASHIFT,ACHEM,ATSHF,CJ1,TOLUP,TOLDWN,TOLCLS
C** $FLAG = .TRUE.
NN2 = 1
DO 17 ITR1 = 1,ITR1
CALL ITRATE ITR2) GO TO 20
IF(NJ.GT.CABS(SCALEW))
TEMP = CABS((TEMP - OLDVR)/TEMP)
OLDVR = TEMP
IF(OLDVR.GT.EPSPVIB) GO TO 17
IF($DOVEC .AND. (NCCNV .EQ. 0)) GO TO 17
NN2 = 1
CALL ITRATE
VREAL = CNEDO / SCALEW
VREAL = VREAL + DBLE(ATSHF)
AREAL = SNGL(VREAL)
RETURN 1
CONTINUE
20 VREAL = CNEDO / SCALEW
VREAL = VREAL + DBLE(ATSHF)
AREAL = SNGL(VREAL)
CALL VECOUT(7,617)
CONTINUE
17 VREAL = CNEDO / SCALEW

```

```

38050
38060
38070
38080
38090
38100
38110
38120
38130
38140
38150
38160
38170
38180
38190
38200
38210
38220
38230
38240
38250
38260
38270
38280
38290
38300
38310
38320
38330
38340
38350
38360
38370
38380
38390
38400
38410
38420
38430
38440
38450
38460

```



```

VREAL = VREAL + DBLE(ATSHF)
AREAL = SNGL(VREAL)
CALL VECOUT(8,619)
19 RETURN
END

```

```

SUBROUTINE ITRATE
C ***
C THIS SUBROUTINE PERFORMS THE ITERATION COMPUTATIONS. POTTERS,
C SCHEME IS USED (EQUATION 29A AND 29B OF REF. 1). THE RAYLEIGH
C QUOTIENT IS CALCULATED AND THE VECTOR SCALED. IN
C ADDITION IF THE VECTOR CONVERGENCE TEST IS TO BE PERFORMED IT IS
C PERFORMED IN THIS SUBROUTINE.
C ***
C IMPLICIT LOGICAL*1 ($),PSAIT,CNEDC,PTFIVE,SCALEW,SCALEX,SCALE3,
C REAL*8 EPSVIB,EPSVEC,SUM1,TEMPW,TEMPX,TWOPI,VREAL,VIMAG,WV,XV,YV,
C 1 SMALL,SUM,ZV,TEMPY,TEMPY,WT,XT,YT
C 2 ZEROCD,ZI,ZV,TEMPY,TEMPY,WT,XT,YT
C COMMON /BL3/ N0,M1,M2,M3
C COMMON /BL4/ NMAX,KL,IBCFNL,IBCFNL,IBCFNL
C COMMON /BL5/ IBCINL,IBCFNL,IBCFNL,IBCFNL,IBCFNL
C COMMON /BL12/ KMAX1,KMAX2,NCONV
C COMMON /BL1/ A(4,4),SEE(4,4),C(4,4)
C COMMON /BL4/ P(4,4,200),ZF1M(4,4,10),ZF2M(4,4,10),ZF3M(4,4,10),
C 1 COMMON/BL24/DG(4,4,4),RAGE(100,4),TOP,BOTTOM,DELSHF,TOLSHF,RZ(12)
C COMMON/BL34/ DCE(4,4,200),DST(4,4,200)
C COMMON/BL104/ WV(4,4,203),XV(4,4,203),YV(4,4,203),FILL(28),SHFTPT(3,100)
C COMMON/BLK1/ VMAS,N2,PVIBES
C COMMON/BLK3/ TWCP1,IBCFNL,IBCFNL,IBCFNL,IBCFNL
C COMMON/BLK4/ ZEROCD,PTFIVE,ONEDO,SMALL,VREAL,VIMAG,AREAL
C COMMON/BLK5/ $ELAG,$TAG,$COVEC
C COMMON/BLK6/ NJ,NK,I4,IGOLD,NUMBER,ITWD
C COMMON/BLK7/SCALEX,SCALEW,SCALE3
C COMMON/BLK8/ EPSVIB,EPSVEC,EPSAIT
C DIMENS:CN,ZV(4,200),DIAG(4)
C 4 EQUIVALENCE CL(4,4),CL2(4,4),WT(812),YT(812)
C 3 (FILL(1),ZT(1)),(FILL(9),ST(1)),(CLO(1,1),ZF1M(1,1,1))
C 2 ((CL1(1,1),ZF2M(1,1,1)),(CL2(1,1),ZF3M(1,1,1)),
C (WV(1,1),WT(1)),(XV(1,1),YV(1,1)),YT(1))
C ***
C INCREMENT THE ITERATION COUNTERS.
C ***
C NJ = NJ + 1
C NK = NK + 1
C ***

```





```

C RETAIN THE LAST TWO SOLUTIONS.
SCALE3 = SCALEX
SCALEX = SCALEW
DO 102 I = IGO, ISTOP
102 Y(I) = XT(I)
DO 103 I = IGO, ISTOP
103 XT(I) = WT(I)
C MULTIPLY THE RIGHT HAND SIDE VECTOR BY THE DIAGONAL
C OF THE MASS MATRIX (W(J,203), J=1,4).
DO 104 J = IBEGIN, IEND
104 W(J,I) = W(J,I) * W(J,203)
C BEGIN FORWARD PASS.
C INITIAL POLE BOUNDARY CONDITIONS ON ZV ARE ZV(J,1)=0 FOR J=1,4.
C IF (BCINL.GE. 0) GO TO 106
DO 105 J = 1,4
105 ZV(J,1) = ZERO
106 DO 3 I = 1,4
SUM = ZERO
ZT(I) = W(I,2)
DO 2 J = 1,4
2 SUM = SUM + DBLE(DG(I,J,1)) * W(J,2)
3 ZV(I,1) = SUM
C FORWARD PASS LOOP.
DO 4 K = 2, KL
KM1 = K - 1
DO 6 I = 1,4
SUM = ZERO
DO 5 J = 1,4
5 SUM = SUM + DBLE(DEE(I,J,K)) * W(J,KP1) * ZV(J,KM1)
6 ZV(I,K) = SUM
7 CONTINUE
C COMPLETE FORWARD PASS AND BEGIN BACKWARD PASS.
C FINAL POLE BOUNDARY CONDITIONS
C ARE HANDLED IN STATEMENTS 20 THRU 29.

```





```

C** IF(IBCENL.LT. 3) GO TO 23
C**
C** KM = KMAX
C** DO 9 I = 1,4
C** SUM = ZERO
C** DO 8 J = 1,4
C** 8 SUM = SUM + DBLE(DEE(I,J,KM)) * WV(J,KMAX1) - DBLE(DST(I,J,KM)) * ZV(J,KL)
C** 9 ZV(I,KMAX) = SUM
C**
C** DO 11 I = 1,4
C** SUM = ZERO
C** DO 10 J = 1,4
C** SUM = SUM + DBLE(ZF2M(I,J,1)) * ZV(J,KMAX)
C** 10 SUM = SUM + DBLE(ZF3M(I,J,1)) * ZV(J,KL)
C** 11 WV(I,KMAX2) = SUM
C** ISTART = 1
C**
C** BACKWARD PASS LOOP.
C**
C** 12 DO 15 KK = ISTART,KMAX
C** K = KMAX1 - KK
C** KP1 = K + 1
C** KP2 = K + 2
C** DO 14 I = 1,4
C** SUM = ZERO
C** DO 13 J = 1,4
C** 13 SUM = SUM - DBLE(P(I,J,K)) * WV(J,KP2)
C** 14 WV(I,KP1) = SUM
C** 15 CONTINUE
C**
C** COMPLETE BACKWARD PASS ( COMPLETES SOLUTION VECTOR WV ).
C**
C** IF(IBCENL.LT. 0) GO TO 30
C**
C** CALL EFG(1,1)
C** CALL ABC
C** DO 18 I = 1,4
C** SUM = ZERO
C** DO 17 J = 1,4
C** 17 SUM = SUM - DBLE(A(I,J)) * WV(J,3) - DBLE(BEE(I,J)) * WV(J,2)
C** 18 ST(I) = SNGL(SUM + ZT(I))
C** CALL MATINV(C,4,ST,1,DETERM,IPIVOT,INDEX,4,ISCALE)
C** DO 19 I = 1,4
C** WV(I,1) = DBLE(ST(I))
C** 19 ZT(I) = ZERO
C** GO TO 30
C**

```



```

C      THE FOLLOWING BLOCK OF STATEMENTS DETERMINE WV(J,KMAX1)
C      FOR A SHELL WITH A FINAL POLE.
C      **
C      **
C      **
20  DO 201 J = 1,4
201  WV(J,KMAX1) = ZEROCD
    IF(M2.EQ.0) GO TO 23
    DO 22 I = 1,4
    SUM = ZEROCD
    DO 21 J = 1,4
    SUM = SUM + DBLE(CL2(I,J)) * ZV(J,KL)
21  SUM = SUM + DBLE(CL2(I,J)) * ZV(J,KL)
22  WV(I,KMAX1) = SUM
23  IF(M1.EQ.0) GO TO 26
    DO 25 I = 1,4
    SUM = ZEROCD
    DO 24 J = 1,4
    SUM = SUM + DBLE(CL1(I,J)) * ZV(J,KL)
24  SUM = SUM + DBLE(CL1(I,J)) * ZV(J,KL)
25  WV(I,KMAX1) = SUM
26  IF(M0.EQ.0) GO TO 29
    DO 28 I = 1,4
    SUM = ZEROCD
    DO 27 J = 1,4
    SUM = SUM + DBLE(CLO(I,J)) * ZV(J,KL)
27  SUM = SUM + DBLE(CLO(I,J)) * ZV(J,KL)
28  WV(I,KMAX1) = SUM
29  ISTART = 2
    GO TO 12
C      **
C      **
C      **
C      CALCULATE EIGENVALUE AND SCALE SOLUTION VECTOR.
C      **
C      **
C      **
30  TEMPV = ZEROCD
    IF($FLAG) GO TO 302
C      **
C      **
C      **
C      IF $FLAG IS FALSE ITRATE CALLED IN ITCPLX.
C      DETERMINE AND SCALE BY THE MAXIMUM(ABSOLUTE) ELEMENT AND RETURN.
C      **
C      **
C      **
    DO 301 I = 160,1STOP
    TEMPW = WT(I)
    TEMPX = DABS(TEMPW)
    IF(TEMPX.LE. TEMPV) GO TO 301
    TEMPV = TEMPX
    TEMPW = TEMPV
301  CONTINUE
    SCALEW = TEMPV
    TEMPW = ONECD / TEMPV
    GO TO 31
C      **
C      **
C      **
C      IF $FLAG IS TRUE ITRATE CALLED IN ITREAL,RANGER,BTTR CR TOPPER.
C      CALCULATE RAYLEIGH QUOTIENT EIGENVALUE AND

```



```

C *** DETERMINE THE MAXIMUM(ABSOLUTE) ELEMENT. ***
302 SUM = ZERO
DO 303 I = 1,4
  TEMPW = WV(J,I) * WV(J,I)
  SUM = SUM + TEMPW
  IF(TEMPW .GT. TEMPV) GO TO 303
  TEMPV = TEMPW
303 CCCONTINUE
  SCALEW = SUM / TEMPV
  IF NN2 = 2 SCALE BY MAXIMUM ELEMENT AND RETURN.
  GO TO (304,31),NN2
  IF NN2 = 1 SOLUTION HAS CONVERGED TO A REAL EIGENVALUE.
  SCALE VECTOR BY RAYLEIGH QUOTIENT. LAST SOLUTION ( THE
  COMPUTE DIFFERENCE BETWEEN THIS AND NEXT STARTING VECTOR FOR THE NEXT EIGENVALUE ). RETURN.
  TEMPW = SUM / SUM
  NN2 = 2
  VIMAG = ZERO
  DO 305 I = 1,4, ISTOP
    WT(I) = WT(I) * TEMPW
    YT(I) = YT(I) - WT(I)
  305 RETURN
31 NCGNV = 1
  VIMAG = ZERO
  SCALE VECTOR BY THE MAXIMUM ELEMENT.
  DO 311 I = 1,4, ISTOP
    WT(I) = WT(I) * TEMPW
  311 WT(I) = WT(I)
  THE FOLLOWING BLOCK OF STATEMENTS PERFORM $DOVEC IS TRUE.
  TEST ON THE EIGENVECTOR. EXECUTED WHEN $DOVEC IS TRUE.
  IF(.NOT.$DOVEC) RETURN

```





```

C *** CALCULATE CONVERGENCE CRITERION. ***
C *** DO 32 I = 1,BEGIN,IEND ***
C *** DO 32 J = 1,4 ***
C *** ZT(J) = CMAX1(DABS(WV(J,I)),ZT(J)) ***
C *** 22 CONTINUE ***
C *** DO 33 I = 1,4 ***
C *** ZT(I) = EPSVEC *** ZT(I) ***
C *** PERFORM THE TEST. ***
C *** DO 34 I = 1,BEGIN,IEND ***
C *** DO 34 J = 1,4 ***
C *** TEMPW = DABS(WV(J,I)) ***
C *** IF(NCONV.EQ.0 .OR. TEMPW.LT.(1.D-03)) GO TO 34 ***
C *** TEMPX = DABS(TEMPW - DABS(XV(J,I))) ***
C *** IF(TEMPX .GE. ZT(J)) GO TO 35 ***
C *** 34 CONTINUE ***
C *** PASSED VECTOR CONVERGENCE TEST. ***
C *** RETURN ***
C *** FAILED VECTORS CONVERGENCE TEST. ***
C *** 35 NCONV = 0 ***
C *** RETURN ***
C *** END ***

SUBROUTINE ITCPLX(*)
C *** IN THE EVENT OF NO CONVERGENCE TO A REAL EIGENVALUE IN THE SUB- ***
C *** SPECIFIED NUMBER OF ITERATIONS CONTROL IS PASSED TO THIS SUB- ***
C *** ROUTINE. THE ALGORITHM ATTEMPTS TO FIND THE TWO EIGENVALUES ***
C *** CLOSEST TO THE SHIFT POINT. THE METHOD OF LEAST SQUARES IS USED ***
C *** TO DETERMINE THE COEFFICIENTS OF THE QUADRATIC EQUATION WHOSE ***
C *** ASSUMED ROOTS ARE THE EIGENVALUES SOUGHT. THEY MAY BE REAL OR A ***
C *** COMPLEX CONJUGATE PAIR. ***
C *** IMPLICIT LOGICAL #1 ($), #2 (D) ***
C *** REAL #8 ***
C *** ISCALEW, SCALEX, SCALEY, P, Q, U, WV, XI, YI, INCPI, ***
C *** 28, C, D, E, F, G, H, TWOPI, IPEGIN, IEND, IGO, ISTOP, ***
C *** COMMON /BLK3/ ZERO, TWOPI, IPEGIN, IEND, IGO, ISTOP, ***
C *** COMMON /BLK4/ ZERODG, $TAG, $DCCVEC ***
C *** COMMON /BLK5/ $FLAG, $TAG, $DCCVEC ***
C *** COMMON /BLK6/ NJ, NK, I4, IRGOLD, NUMBER, ITWO ***

```





41310  
41320  
41330  
41340  
41350  
41360  
41370  
41380  
41390  
41400  
41410  
41420  
41430  
41440  
41450  
41460  
41470  
41480  
41490  
41500  
41510  
41520  
41530  
41540  
41550  
41560  
41570  
41580  
41590  
41600  
41610  
41620  
41630  
41640  
41650  
41660  
41670  
41680  
41690  
41700  
41710  
41720  
41730  
41740  
41750  
41760  
41770  
41780

```

COMMON/BLK7/SCALEX,SCALEW,SCALEZ
COMMON/BLK8/EP$VIB,EP$VEC,EP$AIT
COMMON/BLK9/ITR1,ITR2
COMMON/BLK10/ASHIFT,ASHFUM,ATSHF,BII,TOLUP,TOLDWN,TOLCLS
COMMON/BL104/WV(4,203),XV(4,203),YV(4,203),SHFTPT(3,100)
DIMENSION WT(812),XT(812),YT(812)
EQUIVALENCE (WV(1,1),WT(1)),(XV(1,1),XT(1)),(YV(1,1),YT(1))
C*****
CALL START(1)
OLDVR = ZERODD
OLDVI = ZERODD
SCALEW = CNEODD
DO 5 I1 = 1,ITR1
$FLAG = .FALSE.
CALL ITRATE
B = ZERODD
C = ZERODD
D = ZERODD
F = ZERODD
G = ZERODD
DO 1 I = 1, I STOP IGC, I STOP
1 B = B + XT(I) * XT(I)
2 C = C + XT(I) * XT(I)
3 D = D + XT(I) * XT(I)
4 F = F + XT(I) * XT(I)
9 G = G + XT(I) * XT(I)
B = B * SCALEX * SCALEW
C = C * SCALEX * SCALEW
D = D * SCALEX * SCALEW
F = F * SCALEX * SCALEW
G = G * SCALEX * SCALEW
H = F - E * U
IF(H.EQ.ZERODD) H = SMALL
ZC = D/B
C = (G-E*ZC)/H
P = ZC - U*Q
IF(Q.EQ.ZERODD) Q = -SMALL
ZB = CNEODD/Q
ZA = -P*ZB
VREAL = PTFIVE * ZA
VRO = ZA + 4*D * ZB
IF(RAD*GE*ZERODD) $FLAG = .TRUE.
IF(RAD*LT*ZERODD) RAD = -RAD
VIMAG = PTFIVE * DSCRT(RAD)

```



41790  
41800  
41810  
41820  
41830  
41840  
41850  
41860  
41870  
41880  
41890  
41900  
41910  
41920  
41930  
41940  
41950  
41960  
41970  
41980  
41990

```
IF(DABS(VREAL)-LT.(1.D-13)) VREAL = SMALL
IF(CABS(VIMAG)-LT.(1.D-13)) VIMAG = SMALL
DELVR = CABS((CABS(VREAL)-CABS(OLDCR))/VREAL)
OLDCR = VREAL
DELVI = CABS((CABS(VIMAG)-CABS(OLDVI))/VIMAG)
OLDVI = VIMAG
VREAL = VREAL + DBLE(ATSHF)
IF(NJ.GT.ITR2) CALL VECOUT(16,85)
IF(DELVR.GT.EPSVIB) GO TO 5
IF(DELVI.GT.EPSVIB) GO TO 5
IF($FLAG) GO TO 8
CALL VECOUT(18,86)
5 CONTINUE
CALL VECOUT(16,87)
8 TEMP = VIMAG
VIMAG = VREAL - TEMP
VREAL = VREAL + TEMP
CALL VECOUT(17,86)
6 RETURN
7 END
```

42000  
42010  
42020  
42030  
42040  
42050  
42060  
42070  
42080  
42090  
42100  
42110  
42120  
42130  
42140  
42150  
42160  
42170  
42180  
42190  
42200  
42210  
42220  
42230  
42240

```
SUBROUTINE INITAA
C *** THIS SUBROUTINE SETS THE OPERATING PARAMETERS ***
C *** OF THE PROGRAM FOR THE VIBRATION ANALYSES. ***
C ***
C *** IMPLICIT LOGICAL I($), DNEO, SMALL, VREAL, VIMAG, EPSVIB, EPSAIT, ***
C *** REAL*8 ZERO, PTFIVE, CNEDO, SMALL, VREAL, VIMAG, EPSVIB, EPSAIT, ***
C *** TWOPI, WV, XV, YV ***
1 COMMON /IBL1/ MNMAX
COMMON /IBL2/ N(10), MNINIT
COMMON /IBL4/ KMAX, KL
COMMON /IBL5/ IBCINL, IBCFNL
COMMON /IBL7/ MNMAXO, MAXD(10), MAXS(10), MAXSV(10), MODVIB(3,100), ITEM
IP(110)
COMMON /IBL9/ MAXM
COMMON /IBL10/ IFEREC, NTHMAX
COMMON /IBL12/ KMAX1, KMAX2, NCONV
COMMON /IBL13/ ITRMAX, LCMAX
COMMON /IBL6/ SOE, OSE, ALCAD
COMMON /IBL16/ EPS
COMMON /BL24/ DG(4,4,4), RANGE(100,4), TOP, BOTTOM, DELSHF, TOLSHF, RZ(12)
COMMON /BL100/ TEEC, $DYRMC
COMMON /BL102/ DELOAD
COMMON /BL104/ WV(4,203), XV(4,203), YV(4,203), FILL(28), SHFTPT(3,100)
COMMON /BLK1/ VMAS, NK2, $VIRIES
```



```

42250 COMMON /BLK3/ TWOPI,IBEGIN,IEND,IGC,ISTOP
42260 COMMON /BLK4/ ZEROODC,PTFIVE,ONEDO,SMALL,VREAL,VIMAG,AREAL
42270 COMMON /BLK8/ EPSVIB,EPSVIB2,EPSSVEC,EPSSAIT
42280 COMMON /BLK9/ ITR1,ITR2,ASHFUM,ATSHF,BII,TOLUP,TOLDWN,TOLCLS
42290 COMMON /BLK10/ ASHIFT,ASHFUM,ATSHF,BII,TOLUP,TOLDWN,TOLCLS
42300 COMMON /BLK11/ DIAG(4)
42310 COMMON /BLK12/ NTRY
42320 COMMON /BLK13/ DIF1,DIF2,DIF3,DIF4,TOLLOW,TOLHI,TOLRNG
42330 COMMON /BLPLOT/ KLI(29),$PLOTS,$MODAL
42340 COMMON /BLDATA/ TITLE,NC,IMODE,NDIMEN,IPRINT,LCHMAX,IC
42350 DIMENSION TITLE(18)
42360 **
42370 ** THE FOLLOWING BLOCK OF STATEMENTS SETS THE CONSTANTS FOR
42380 ** THE DOUBLE PRECISION PORTION OF THE ITERATION CALCULATIONS.
42390 **
42400 ONEDO = 1.00
42410 PTFIVE = 1.500
42420 SMALL = 1.00E-20
42430 TWOPI = 6.283185307179586400
42440 ZEROODC = 0.00
42450 **
42460 ** THE FOLLOWING BLOCK OF STATEMENTS SETS THE
42470 ** ITERATION AND CONVERGENCE PARAMETERS.
42480 **
42490 EPSSAIT = 1.0E-02
42500 EPSVIB = DBLE(EPS)
42510 IF(EPSVEC.EQ.ZEROODC) EPSSVEC = DBLE(EPS)
42520 IF($PLOTS.AND.(KMAX.GT.100)) KMAX=100
42530 KL = KMAX - 1
42540 KMAX1 = KMAX + 1
42550 KMAX2 = KMAX + 2
42560 IBEGIN = 1
42570 IF(IACINL.LT.0) IBEGIN = 2
42580 IGC = (IBEGIN * 4) - 3
42590 IEND = KMAX2
42600 IF(I8CFNL.LT.0) IEND = KMAX1
42610 ISTOP = IEND * 4
42620 ITR1 = ITRMAX
42630 IF(ITR2.EQ.0) ITR2 = ITR1 - 2
42640 NN2 = 2
42650 DIAG(4) = 0.0
42660 DO 1 I = 1,4
42670 1 WV(I,203) = DBLE(DIAG(I))
42680 **
42690 ** THE FOLLOWING BLOCK OF STATEMENTS SETS THE OPERATING PARAMETERS
42700 ** FOR THE FREQUENCY RANGE SEARCH MODE OF OPERATION.
42710 **
42720 TOLCLS = .4

```















```

201 FORMAT('1',42X,'THIS IS A FREQUENCY RANGE SEARCH FOR HARMONIC',I3)
202 FORMAT('0',24X,'BOTTOM OF RANGE IS',E12.4,' HZ',I2X,'TOP OF RANGE
3 IS',E12.4,' HZ',/35X,'BOTTOM =',E12.4,' HZ',I2X,'TOP =',E12.4)
203 FORMAT('0',41X,'DIAGONAL OF MASS MATRIX IS (',4F3.0,' )')
207 FORMAT('0',43X,'THE CONVERGENCE TEST ON THE VECTOR WILL BE PERFORM
1ED',/50X,'CONVERGENCE CRITERION IS',E10.2)
208 FORMAT('0',38X,'THE CONVERGENCE TEST ON THE VECTOR WILL NOT BE PER
2FORMED')
2 WRITE(6,204) ITRI
2 RETURN 1
204 FORMAT('0',44X,'CHECK PERMISSIBLE NUMBER OF ITERATIONS (',I2,'')
3 WRITE(6,205) EPSVIB
205 FORMAT('0',51X,'EPS IS VERY SMALL (',E9.2,'')
4 WRITE(6,206)
206 RETURN 1
206 FORMAT('0',55X,'CHECK RANGE END POINTS')
5 WRITE(6,219) MODVIB(2,IVB)
6 WRITE(6,203) DIAG
7 WRITE(6,203) DIAG
IF($DOVEC) WRITE(6,207) EPSVEC
IF(.NOT.$DOVEC) WRITE(6,208)
219 FORMAT('1',41X,'THE LOWEST FREQUENCY IN HARMONIC',I4,' IS DETERMIN
6ED')
6 WRITE(6,220) MODVIB(1,IVB),MODVIB(2,IVB)
7 WRITE(6,203) DIAG
IF($DOVEC) WRITE(6,207) EPSVEC
IF(.NOT.$DOVEC) WRITE(6,208)
220 FORMAT('1',41X,I2,' SHIFT POINT(S) FOR HARMONIC',I3,' IS(ARE) PROV
7ICED')
7 WRITE(6,209) NK
8 WRITE(6,213) VREAL
IF(ITEREST.EQ.1) WRITE(6,221)
WRITE(6,224) SCALEW,SCALEX,SCALE3
NJ = 0
209 FORMAT('0',36X,'NO CONVERGENCE TO A REAL EIGENVALUE AFTER ',I5,' I
1TERATIONS')
210 FORMAT('0',38X,'RAYLEIGH QUOTIENT FREQUENCY =',E23.14,' HZ')
221 FORMAT('0',38X,'THE FREQUENCY IS IMAGINARY.')
224 FORMAT('0',34X,'THE LAST NEXT TO NEXT TO NEXT TO LAST EIG
5ENVALUES ARE',/35X,3E21.12)
8 WRITE(6,209) NK
9 WRITE(6,213) VREAL
IF(ITEREST.EQ.1) WRITE(6,221)
WRITE(6,224) SCALEW,SCALEX,SCALE3
2 RETURN 1

```



```

9 WRITE(6,211) NK
  WRITE(6,210) VREAL
  IF(I TEST .EQ. 1) WRITE(6,221)
  DO 91 I = 1GO, ISTOP
    Z(I) = SNGL(WT(I))
    CALL OUTP(IMODE)
    RETURN 1
211 FORMAT('0',47X,'CONVERGED. SOLUTION. ITERATIONS =',I6)
10 WRITE(6,301) BOTTOM
  RETURN 1
301 FORMAT('0',50X,'THE RANGE SIZE HAS BEEN REDUCED.'/51X,'BOTTOM OF R
11 WRITE(6,302)
  RETURN 1
302 FORMAT('0',55X,'RANGE HAS BEEN COVERED')
12 WRITE(6,303) NTRY
  RETURN 1
303 FORMAT('0',38X,'UNABLE TO DETERMINE A MINIMUM FREQUENCY AFTER ',I
13 CONTINUE
  RETURN 1
14 WRITE(6,305) NTRY
  RETURN 1
305 FORMAT('0',38X,'UNABLE TO DETERMINE A MAXIMUM FREQUENCY AFTER ',I
15 WRITE(6,304) TOP
  RETURN 1
304 FORMAT('0',50X,'THE RANGE SIZE HAS BEEN REDUCED.'/51X,'TOP OF RANG
2E IS NOW',E14.6)
16 WRITE(6,212) NK,VREAL,VIMAG
  WRITE(6,225) SCALEW,SCALEX,SCALE3
  NJ = 0
  RETURN 1
212 FORMAT('0',47X,'NO CONVERGENCE AFTER',I6,' ITERATIONS.'/42X,'EIGEN
5VALUES(S) ARE',2E15.6)
225 FORMAT('0',29X,'THE LAST, NEXT TO LAST AND NEXT TO NEXT TO LAST VE
4CTOR ELEMENT MAXIMUMS ARE',/35X,3D21.12)
17 WRITE(6,213)
  WRITE(6,214) NK,VREAL,VIMAG
  IF(I TEST .EQ. 1) WRITE(6,222)
  IF(I TEST .EQ. 2) WRITE(6,223)
  RETURN 1
213 FORMAT('0',45X,'CONVERGED SOLUTION - TWO REAL EIGENVALUES',/21X,'AN
1 INCREASE IN THE CONVERGENCE CRITERIA(EPS) AND/OR AN INCREASE IN T
2HE PERMISSIBLE NUMBER OF',/16X,'ITERATIONS(ITRMAX) IS RECOMMENDED A
3S A POSSIBLE SOLUTION TO THE PROBLEM OF A POORLY CONDITIONED MATRI
4X,')
214 FORMAT('0',29X,'ITERATIONS ARE',I6,10X,'FREQUENCIES, IN HZ, ARE',2

```

44120  
 44130  
 44140  
 44150  
 44160  
 44170  
 44180  
 44190  
 44200  
 44210  
 44220  
 44230  
 44240  
 44250  
 44260  
 44270  
 44280  
 44290  
 44300  
 44310  
 44320  
 44330  
 44340  
 44350  
 44360  
 44370  
 44380  
 44390  
 44400  
 44410  
 44420  
 44430  
 44440  
 44450  
 44460  
 44470  
 44480  
 44490  
 44500  
 44510  
 44520  
 44530  
 44540  
 44550  
 44560  
 44570  
 44580  
 44590





44595  
44600  
44610  
44620  
44630  
44640  
44650  
44660  
44670  
44680  
44690  
44700  
44710  
44720  
44730  
44740  
44750  
44760  
44770

```

8E16.6)
222 FORMAT('0',63X,'THE FIRST FREQUENCY IS IMAGINARY.')
223 FORMAT('0',63X,'BOTH FREQUENCIES ARE IMAGINARY.')
18 WRITE(6,215) NK,VREAL,VIMAG
RETURN 1
215 FORMAT('0',46X,'CONVERGED SOLUTION - COMPLEX EIGENVALUE',/34X,'ITER
1ATIONS ARE',16,10X,'EIGENVALUE IS',4X,2E15.6)
19 WRITE(6,216) NUMBER
RETURN 1
216 FORMAT('0',51X,15,' FREQUENCIES HAVE BEEN FOUND')
20 WRITE(6,217) BOTTOM,RANGE(1,1),RANGE(1,2)
RETURN 1
217 FORMAT('0',47X,'BOTTOM =',E12.4/48X,'LOWEST EIGENVALUE (RANGE(1,1)
2) =',E14.6/48X,'RANGE(1,2) =',E12.4)
21 WRITE(6,218) TOP,RANGE(1,4),RANGE(1,3)
RETURN 1
218 FORMAT('0',47X,'TOP =',E12.4/48X,'HIGHEST EIGENVALUE (RANGE(1,4)
3) =',E14.6/48X,'RANGE(1,3) =',E12.4)
END

```

44780  
44790  
44800  
44810  
44820  
44830  
44840  
44850  
44860  
44870  
44880  
44890  
44900  
44910  
44920  
44930  
44940  
44950  
44960  
44970  
44980  
44990  
45000  
45010  
45020  
45030  
45040

```

SUBROUTINE START(K)
C *****
C SUBROUTINE START INITIALIZES THE EIGENVALUES AND EIGENVECTORS
C PRIOR TO THE START OF EACH ITERATION.
C *****
REAL*8 TWOPI,ZERODD,PTFIVE,ONEDO,SMALL,VREAL,VIMAG,SCALEW,SCALEX,SCALEY,
1SCALEZ,WV,WT,XV,XT,YV,YT,TEMP
COMMON/BLK104/ WV(4,203),XV(4,203),FILL(28),SHFTPT(3,100)
COMMON/BLK3/ TWOPI,18FIVE,IEND,IGD,ISTOP
COMMON/BLK4/ ZERODD,PTFIVE,ONEDO,SMALL,VREAL,VIMAG,AREAL
COMMON/BLK6/NJ,NK,I4,IRGOLD,NUMBER,ITWO
COMMON/BLK7/ SCALEX,SCALEY,SCALEZ
DIMENSION WT(812),XT(812),YT(812)
EQUIVALENCE (WV(1,1),WT(1)),(XV(1,1),XT(1)),(YV(1,1),YT(1))
C *****
NJ = 0
NK = 0
ITWO = 0
SCALEW = 1
SCALEX = 1
SCALEY = 1
SCALEZ = 1
VIMAG = 1
GO TO (1,3,3),K
1 DO 2 I = 1,IGD,I STOP
2 WT(I) = 1
3 TEMP = 1

```





```

45050 4 I = IGC,ISTCP
45051 TEMP = DMXL(DABS(YT(I)),TEMP)
45052 IF(TEMP .LT. (1.C-40)) GO TO 1
45053 TEMP = ONECC / TEMP
45054 DC 5 I = IGC,ISTCP
45055 WT(I) = YT(I) * TEMP
45056 DO 7 I = IGC,ISTCP
45057 XT(I) = CNECC
45058 RETURN
45059 END

```

```

45150 SUBROUTINE MATINV(A,N,B,M,DETERM,IPIVOT,INDEX,NMAX,ISCALE)
45151 *****
45152 THIS SUBROUTINE SOLVES THE MATRIX EQUATION AX=B, WHERE A IS
45153 A SQUARE COEFFICIENT MATRIX AND B IS A MATRIX OF CONSTANT VEC-
45154 TORS.
45155 A(INVERSE) IS ALSO OBTAINED AND THE DETERMINANT OF A IS AVAIL-
45156 ABLE.
45157 THE FOLLOWING MUST BE DIMENSIONED IN THE CALLING PROGRAM:
45158 IPIVOT(N MAX), INDEX(N MAX,2), A(N MAX,N MAX), B(N MAX,N MAX)
45159 WHERE:
45160 A = NAME OF 2-DIMENSIONAL ARRAY TO BE INVERTED
45161 N = ORDER OF A (N<=NMAX)
45162 B = NAME OF 2-DIMENSIONAL ARRAY TO BE MULTIPLIED
45163 BY A(INVERSE)
45164 M = NUMBER OF COLUMN VECTORS IN B
45165 NOTE: M = 0 SIGNALS INVERSION ONLY)
45166 IPIVOT = TEMPORARY STORAGE BLOCK
45167 INDEX = TEMPORARY STORAGE BLOCK
45168 NMAX = MAXIMUM ORDER OF A (AS DIMENSIONED IN THE
45169 CALLING PROGRAM)
45170 DETERM = VALUE OF DETERMINANT AS GIVEN BELOW
45171 ISCALE = USED IN FORMULA BELOW
45172 DETERMINANT(A) = (10**18) * ISCALE*(DETERM)

```

```

45400 A(INVERSE) IS STORED IN A
45401 A(INVERSE) IS STORED IN B
45402 *****
45403 DIMENSION IPIVOT(N), A(NMAX,N), B(NMAX,M), INDEX(NMAX,2)
45404 EQUIVALENCE (IROW,JROW), (ICOL,JCOL), (AMAX,I,SWAP)
45405 *****
45406 INITIALIZATION
45407 *****

```

```

45408 5 ISCALE=0
45409 6 RI=10.0**18
45410 7 R2=1.0/R1

```



```

10 CTERM=1.0
15 DO 20 J=1,N
20 IPIVOT(J)=0
30 DO 550 I=1,N
    SEARCH FOR PIVOT ELEMENT
40 AMAX=0.0
45 DO 105 J=1,N
50 IF (IPIVOT(J)-1) 60, 105, 60
60 DO 100 K=1,N
70 IF (IPIVOT(K)-1) 80, 100, 740
80 IF (ABS(AMAX)-ABS(A(J,K)))85, 100, 100
85 IROW=J
90 ICOLUM=K
95 AMAX=A(J,K)
100 CONTINUE
110 IPIVOT(ICOLUM)=IPIVOT(ICOLUM)+1
    INTERCHANGE ROWS TO PUT PIVOT ELEMENT ON DIAGONAL
130 IF (IROW-ICOLUM) 140, 260, 140
140 DETERM=-CTERM
150 DO 200 L=1,N
160 SWAP=A(IROW,L)
170 A(IROW,L)=A(ICOLUM,L)
200 A(ICOLUM,L)=SWAP
205 IF (M) 260, 260, 210
210 DO 250 L=1,M
220 SWAP=B(IROW,L)
230 B(IROW,L)=B(ICOLUM,L)
250 B(ICOLUM,L)=SWAP
260 INDEX(I,1)=IROW
270 INDEX(I,2)=ICOLUM
310 PIVOT=A(ICOLUM,ICOLUM)
    SCALE THE DETERMINANT
1000 PIVOTI=PIVOT
1005 IF (ABS(DETERM)-R1) 1030, 1010, 1010
1010 DETERM=DETERM/R1
    ISCALE=ISCALE+1
1020 IF (ABS(DETERM)-R1) 1060, 1020, 1020
    DETERM=DETERM/R1
    ISCALE=ISCALE+1
    GO TO 1060
1030 IF (ABS(DETERM)-P2) 1040, 1040, 1060

```



```

1040 DETERM=DETERM*R1
1050 ISCALE=ISCALE-1
1060 IF(ABS(DETERM)-R2)1050,1060
1070 DETERM=DETERM*R1
1080 ISCALE=ISCALE-1
1090 IF(ABS(PIVOTI)-R1)1050,1070
1100 PIVCTI=PIVCTI/R1
1110 ISCALE=ISCALE+1
1120 IF(ABS(PIVOTI)-R1)1050,1080
1130 PIVCTI=PIVCTI/R1
1140 ISCALE=ISCALE+1
1150 GO TO 320
1160 IF(ABS(PIVOTI)-R2)2000,2000,320
1170 PIVCTI=PIVCTI*R1
1180 ISCALE=ISCALE-1
1190 IF(ABS(PIVOTI)-R2)2010,2010,320
1200 PIVOTI=PIVOTI*R1
1210 ISCALE=ISCALE-1
1220 DETERM=DETERM*PIVOTI
1230
1240 C DIVIDE PIVOT ROW BY PIVOT ELEMENT
1250
1260
1270
1280
1290
1300
1310
1320
1330 A(ICOLU,ICOLU)=1.0
1340 DO 350 L=1,N
1350 A(ICOLU,L)=A(ICOLU,L)/PIVOT
1360 IF(N) 380, 360
1370 DO 370 L=1,M
1380 B(ICOLU,L)=B(ICOLU,L)/PIVOT
1390
1400
1410
1420
1430
1440
1450
1460
1470
1480
1490
1500
1510
1520
1530
1540
1550
1560
1570
1580
1590
1600
1610
1620
1630
1640
1650
1660
1670
1680
1690
1700
1710
1720
1730
1740
1750
1760
1770
1780
1790
1800
1810
1820
1830
1840
1850
1860
1870
1880
1890
1900
1910
1920
1930
1940
1950
1960
1970
1980
1990
2000
2010
2020
2030
2040
2050
2060
2070
2080
2090
2100
2110
2120
2130
2140
2150
2160
2170
2180
2190
2200
2210
2220
2230
2240
2250
2260
2270
2280
2290
2300
2310
2320
2330
2340
2350
2360
2370
2380
2390
2400
2410
2420
2430
2440
2450
2460
2470
2480
2490
2500
2510
2520
2530
2540
2550
2560
2570
2580
2590
2600
2610
2620
2630
2640
2650
2660
2670
2680
2690
2700
2710
2720
2730
2740
2750
2760
2770
2780
2790
2800
2810
2820
2830
2840
2850
2860
2870
2880
2890
2900
2910
2920
2930
2940
2950
2960
2970
2980
2990
3000
3010
3020
3030
3040
3050
3060
3070
3080
3090
3100
3110
3120
3130
3140
3150
3160
3170
3180
3190
3200
3210
3220
3230
3240
3250
3260
3270
3280
3290
3300
3310
3320
3330
3340
3350
3360
3370
3380
3390
3400
3410
3420
3430
3440
3450
3460
3470
3480
3490
3500
3510
3520
3530
3540
3550
3560
3570
3580
3590
3600
3610
3620
3630
3640
3650
3660
3670
3680
3690
3700
3710
3720
3730
3740
3750
3760
3770
3780
3790
3800
3810
3820
3830
3840
3850
3860
3870
3880
3890
3900
3910
3920
3930
3940
3950
3960
3970
3980
3990
4000
4010
4020
4030
4040
4050
4060
4070
4080
4090
4100
4110
4120
4130
4140
4150
4160
4170
4180
4190
4200
4210
4220
4230
4240
4250
4260
4270
4280
4290
4300
4310
4320
4330
4340
4350
4360
4370
4380
4390
4400
4410
4420
4430
4440
4450
4460
4470
4480
4490
4500
4510
4520
4530
4540
4550
4560
4570
4580
4590
4600
4610
4620
4630
4640
4650
4660
4670
4680
4690
4700
4710
4720
4730
4740
4750
4760
4770
4780
4790
4800
4810
4820
4830
4840
4850
4860
4870
4880
4890
4900
4910
4920
4930
4940
4950
4960
4970
4980
4990
5000
5010
5020
5030
5040
5050
5060
5070
5080
5090
5100
5110
5120
5130
5140
5150
5160
5170
5180
5190
5200
5210
5220
5230
5240
5250
5260
5270
5280
5290
5300
5310
5320
5330
5340
5350
5360
5370
5380
5390
5400
5410
5420
5430
5440
5450
5460
5470
5480
5490
5500
5510
5520
5530
5540
5550
5560
5570
5580
5590
5600
5610
5620
5630
5640
5650
5660
5670
5680
5690
5700
5710
5720
5730
5740
5750
5760
5770
5780
5790
5800
5810
5820
5830
5840
5850
5860
5870
5880
5890
5900
5910
5920
5930
5940
5950
5960
5970
5980
5990
6000
6010
6020
6030
6040
6050
6060
6070
6080
6090
6100
6110
6120
6130
6140
6150
6160
6170
6180
6190
6200
6210
6220
6230
6240
6250
6260
6270
6280
6290
6300
6310
6320
6330
6340
6350
6360
6370
6380
6390
6400
6410
6420
6430
6440
6450
6460
6470
6480
6490
6500
6510
6520
6530
6540
6550
6560
6570
6580
6590
6600
6610
6620
6630
6640
6650
6660
6670
6680
6690
6700
6710
6720
6730
6740
6750
6760
6770
6780
6790
6800
6810
6820
6830
6840
6850
6860
6870
6880
6890
6900
6910
6920
6930
6940
6950
6960
6970
6980
6990
7000
7010
7020
7030
7040
7050
7060
7070
7080
7090
7100
7110
7120
7130
7140
7150
7160
7170
7180
7190
7200
7210
7220
7230
7240
7250
7260
7270
7280
7290
7300
7310
7320
7330
7340
7350
7360
7370
7380
7390
7400
7410
7420
7430
7440
7450
7460
7470
7480
7490
7500
7510
7520
7530
7540
7550
7560
7570
7580
7590
7600
7610
7620
7630
7640
7650
7660
7670
7680
7690
7700
7710
7720
7730
7740
7750
7760
7770
7780
7790
7800
7810
7820
7830
7840
7850
7860
7870
7880
7890
7900
7910
7920
7930
7940
7950
7960
7970
7980
7990
8000
8010
8020
8030
8040
8050
8060
8070
8080
8090
8100
8110
8120
8130
8140
8150
8160
8170
8180
8190
8200
8210
8220
8230
8240
8250
8260
8270
8280
8290
8300
8310
8320
8330
8340
8350
8360
8370
8380
8390
8400
8410
8420
8430
8440
8450
8460
8470
8480
8490
8500
8510
8520
8530
8540
8550
8560
8570
8580
8590
8600
8610
8620
8630
8640
8650
8660
8670
8680
8690
8700
8710
8720
8730
8740
8750
8760
8770
8780
8790
8800
8810
8820
8830
8840
8850
8860
8870
8880
8890
8900
8910
8920
8930
8940
8950
8960
8970
8980
8990
9000
9010
9020
9030
9040
9050
9060
9070
9080
9090
9100
9110
9120
9130
9140
9150
9160
9170
9180
9190
9200
9210
9220
9230
9240
9250
9260
9270
9280
9290
9300
9310
9320
9330
9340
9350
9360
9370
9380
9390
9400
9410
9420
9430
9440
9450
9460
9470
9480
9490
9500
9510
9520
9530
9540
9550
9560
9570
9580
9590
9600
9610
9620
9630
9640
9650
9660
9670
9680
9690
9700
9710
9720
9730
9740
9750
9760
9770
9780
9790
9800
9810
9820
9830
9840
9850
9860
9870
9880
9890
9900
9910
9920
9930
9940
9950
9960
9970
9980
9990

```



```

6400 JCOLUMN=INDEX(L,2)
6500 DO 705 K=1,N
6600 SWAP=A(K,JROW)
6700 A(K,JROW)=A(K,JCOLUMN)
7000 A(K,JCOLUMN)=SWAP
705 CONTINUE
710 CONTINUE
740 RETURN
END

```

SUBROUTINE FLYNIV

[illegible]













## LIST OF REFERENCES

1. National Aeronautics and Space Administration Report NASA CR-1987, A Computer Program for the Geometrically Nonlinear Static and Dynamic Analysis of Arbitrarily Loaded Shells of Revolution, Theory and Users Manual, by Robert E. Ball, April 1972.
2. National Aeronautics and Space Administration Report NASA CR-909, A Geometrically Nonlinear Analysis of Arbitrarily Loaded Shells of Revolution, by Robert E. Ball, January, 1968.
3. Ryan, B.A., A Digital Computer Study of the Buckling of Actual Imperfect Cylinders - A Modification of Program SATANS - Theory of User's Manual for SATANS-I and SATANS-II, Ae.E. Thesis, Department of Aeronautics, Naval Postgraduate School, Monterey, California, Dec. 1972.
4. Sanders, J. Lyell, Jr., "Nonlinear Theories for Thin Shells", Quarterly Journal of Applied Mathematics, vol. 21, no. 1, p. 21-36, 1963.
5. Budiansky, B. and Radkowski, P., "Numerical Analysis of Unsymmetrical Bending of Shells of Revolution", Journal of the American Institute of Aeronautics and Astronautics, vol. 1, no. 8, p. 1833-1842, August, 1963.
6. Air Force Flight Dynamics Laboratory Technical Report AFFDL-TR-68-144, Static, Free Vibration, and Stability Analysis of Thin, Elastic Shells of Revolution, part II, by Arturs Kalnins, March, 1969.
7. Wilkinson, J.H., The Algebraic Eigenvalue Problem, Clarendon Press, 1965.
8. Bodewig, E., Matrix Calculus, 2nd ed., Part IV, North-Holland, 1959.
9. Froberg, Carl-Erik, Introduction to Numerical Analysis, p. 93-103, 2nd ed., Addison-Wesley, 1969.
10. Crandall, S.H., Ph.D., Engineering Analysis, Chaps. 4,5, McGraw-Hill, 1956.
11. Fox, L., An Introduction to Numerical Linear Algebra, Chap. 8, Oxford University Press, 1965





12. Meirovitch, L., Analytical Methods in Vibrations, Chap. 4, MacMillan Company, 1967.
13. Prentis, J.M. and Leckie, F.A., Mechanical Vibrations: An Introduction to Matrix Methods, Chap. 3, Longmans, Green and Company, Ltd., 1963.
14. Anderson, M.S., Fulton, R.E., Heard, W.L., Jr., and Walz, J.E., Stress, Buckling and Vibration Analysis of Shells of Revolution, paper presented at the Conference on Computer Oriented Analysis of Shell Structures, Palo Alto, California, 10 August, 1970.
15. Cohen, G.A., Computer Analysis of Axisymmetric Free Vibrations of Ring-Stiffened Orthotropic Shells of Revolution, Journal of The American Institute of Aeronautics and Astronautics, vol. 3, no. 12, p. 2305-2312, December, 1965.
16. Westlake, J.R., A Handbook of Numerical Matrix Inversion and Solution of Linear Equations, John Wiley and Sons, 1968.
17. Gregory, R.T. and Karney, D.L., A Collection of Matrices for Testing Computational Algorithms, Wiley-Interscience, 1969.
18. Moler, C.B. and Stewart, G.W., An Algorithm for the Generalized Matrix Eigenvalue Problem  $Ax = \lambda Bx$ , Stanford University Computer Science Department (STAN-CS-232-71), August, 1971.
19. INTERNATIONAL MATHEMATICAL AND STATISTICAL LIBRARIES, LIBRARY 1, IMSL, 2nd ed., July, 1972.
20. Kamel, H.A. and Lambert, R.L., Solution of Structural Eigenvalue Problems Using Sparsely Populated Matrices, Paper presented at the Conference on Computer Oriented Analysis of Shell Structures, 10 August, 1970.
21. Fulton, R.E., and Blum, R.E., A Modification of Potters' Method for Solving Eigenvalue Problems Involving Tridiagonal Matrices, Astronautics, vol. 4, no. 12, p. 2231-2232, December, 1966.
22. National Aeronautics and Space Administration Report NASA TN D-3831, Vibration and Buckling of Prestressed Shells of Revolution, by Paul A. Cooper, March, 1967.
23. National Aeronautics and Space Administration Report NASA SP-221, The NASTRAN Theoretical Manual, Sections 9 and 10, edited by Richard H. MacNeal, September 1970.





24. National Aeronautics and Space Administration Report NASA TN D-3844, Matrix Analysis of Longitudinal and Torsional Vibrations in Nonuniform Multibranch Beams, by Robert T. Wingate, February, 1967.
25. National Aeronautics and Space Administration Contractor Report NASA-CR-1316, A Theoretical Analysis of the Free Vibration of Discretely Stiffened Cylindrical Shells With Arbitrary End Conditions, by D. M. Egle and K.E. Soder, Jr. June 1969.
26. United States Naval Postgraduate School Report NPS-57BP7161A, Vibration Analysis of Cylindrical Shells by Several Finite Difference Schemes, by R.E. Ball, June 1971.
27. National Aeronautics and Space Administration Technical Note NASA-TN-D- 4972, A Method for Computation of Vibration Modes and Frequencies of Orthotropic Thin Shells of Revolution Having General Meridional Curvature, by H. M. Adelman, D.S. Catherines and W.C. Walton, Jr, January 1969.
28. National Aeronautics and Space Administration Technical Note NASA-TN-D-2767, Conical Shell Vibrations, by D. H. Platus, April 1965.



INITIAL DISTRIBUTION LIST

	No. Copies
1. Defense Documentation Center Cameron Station Alexandria, Virginia 22314	2
2. Library, Code 0212 Naval Postgraduate School Monterey, California 93940	2
3. Professor Robert E. Ball Department of Aeronautics Naval Postgraduate School Monterey, California 93940	2
4. Chairman, Department of Aeronautics Naval Postgraduate School Monterey, California 93940	1
5. LT Ralph R. Nebiker 115 Malloway Lane Monterey, California 93940	2
6. Professor Lars Åke Samuelson The Aeronautical Research Institute of Sweden P. O. Box 11021 S - 161 11 Bromma 11 Sweden	1
7. Mr. Richard Citerly Anamet Laboratories P. O. Box 831 San Carlos, California 94070	1
8. Dr. Robert Fulton Structural Mechanics Branch NASA - Langley Research Center Hampton, Virginia 23365	1
9. LCDR Bruce A. Ryan Naval Air Systems Command Headquarters (Air-09PB) Washington, DC 20360	1



## DOCUMENT CONTROL DATA - R &amp; D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY (Corporate author) Naval Postgraduate School Monterey, California 93940		2a. REPORT SECURITY CLASSIFICATION Unclassified	
		2b. GROUP	
3. REPORT TITLE Natural Modes and Frequencies of Free Vibration of Shells of Revolution - Analysis of a Shell of Linearly Varying Thickness - SATANS-III: Theory and User's Manual			
4. DESCRIPTIVE NOTES (Type of report and inclusive dates) Master's Thesis; March 1973			
5. AUTHOR(S) (First name, middle initial, last name) Ralph Robert Nebiker			
6. REPORT DATE March 1973		7a. TOTAL NO. OF PAGES 303	7b. NO. OF REFS 28
8a. CONTRACT OR GRANT NO.		9a. ORIGINATOR'S REPORT NUMBER(S)	
b. PROJECT NO.			
c.		9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report)	
d.			
10. DISTRIBUTION STATEMENT  Approved for public release; distribution unlimited.			
11. SUPPLEMENTARY NOTES		12. SPONSORING MILITARY ACTIVITY Naval Postgraduate School Monterey, California 93940	

13. ABSTRACT <p>A digital computer program known as SATANS-III is developed as a modification to the program SATANS-I, incorporating the capability to perform free vibration analyses for the natural frequencies and modes of thin, elastic shells of revolution. The geometric and material properties of the shell may vary along the meridian. SATANS-III retains the SATANS-I capability of performing geometrically nonlinear static and dynamic analyses of arbitrarily loaded shells. The free vibration analysis employs the inverse iteration method with spectral shifts, including the capability to search a specific range of frequencies. Several example problems are solved to illustrate the program's validity and accuracy. A user's manual for SATANS-III is presented. In addition, the problem of handling a large, strongly banded, unsymmetric and occasionally singular stiffness matrix in conjunction with a singular mass matrix is treated. A discussion of several solution procedures with respect to this problem is given.</p>
--





14. KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
SHELLS						
SHELLS OF REVOLUTION						
FREQUENCIES						
NATURAL FREQUENCIES						
NATURAL FREQUENCIES OF SHELLS						
VIBRATION						
FREE VIBRATION						
FREE VIBRATION OF SHELLS						
MODES						
MODES OF VIBRATION						
NATURAL MODES OF VIBRATION						
NATURAL MODES OF VIBRATION OF SHELLS						
STRUCTURAL VIBRATIONS						
FREE VIBRATIONS OF STRUCTURES						
EIGENVALUES						
REAL EIGENVALUES						
UNSYMMETRIC MATRICES						
EIGENVALUES OF UNSYMMETRIC MATRICES						
BUCKLING OF SHELLS						
STRUCTURAL ANALYSIS						
SATANS						
SATANS-III						





Thesis  
N3524  
c.2

Nebiker

143418

Natural modes and frequencies of free vibration of shells of revolution - analysis of a shell of linearly varying thickness - Satans-III: Theory and user's manual.

Thesis  
N3524  
c.2

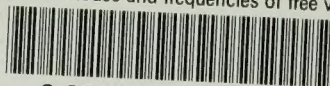
Nebiker

143418

Natural modes and frequencies of free vibration of shells of revolution - analysis of a shell of linearly varying thickness - Satans-III: Theory and user's manual.

thesN3524

Natural modes and frequencies of free vi



3 2768 002 01775 8

DUDLEY KNOX LIBRARY

C.2